CJ-801单板机

上海长江电子计算机厂

总 目 录



CJ801

Z80单板计算机使用手册

目　　录

第一章 概 述

1.1 引 言

近年来，微处理器/微型计算机的发展十分迅速。它的应用已深入到工业、农业、国防、科研、教育、管理以及日常生活(家用电器、玩具)等各个领域。在自动控制和仪器仪表方面的应用尤为突出。随着大规模集成电路的发展，微处理器/微型计算机必将对计算机工业和计算机应用产生深远的影响。

为了推广微型计算机在国民经济各个领域中的应用，研制了一种新型的价格便宜而性能优良的 CJ801—Z80 单板计算机。该机是使用 Z80 系列器件，做在一块印刷电路板上的完整的微型计算机。它结构简单，布局合理，功能齐全，用途广泛。

CJ801—Z80 单板计算机作为“智能”部件，可用于生产过程控制、各种仪器和仪表或机械的单机控制、数据处理等等。它既可独立应用在小型自动控制系统中，又可用于分布系统中的前沿控制。该机尤其适合于初学者学习微型计算机的硬件、指令系统、编写程序的方法和技巧。因此，对大专院校学生和各行各业需要应用微型计算机的科技工作者来说，CJ801—Z80 单板计算机也是一种经济实用的实验教学设备。

CJ801—Z80 单板计算机还具有简易的开发功能，如在用户程序内可设置多至五个的断点，可单步执行存于 RAM 或 PROM 中的程序，可对2716/2758EPROM 进行编程等。因而，可以将它用作调试样机，也可直接将它用于微型机化(μp—based)产品中。

CJ801 作为 CJ80 系列的基础部件，在 S100 总线插座上附加元器件扩展板后，内存容量将增大 16K，可使用 BASIC 语言，并增配字符键盘、显示器(或以黑白电视器代用)、微型打印机等外围设备，这样就构成一种经济实用、功能更为完善的简易微型计算机系统。

1.2 主要技术特性

1. 中央处理单元为 Z80—CPU。

2. 时钟(φ)频率为 1.9968MHz，便于使用 8080A 的接口电路。晶体振荡器的频率为 3.9936MHz。

3. RAM 为 4K 字节的 2114 静态读写存储器。

4. ROM 为 2K 字节，编入监控程序 CJBUG。

5. PROM 插座两个。可插入 4K 字节的 PROM 或 EPROM。

6. Z80—PIO 并行 I/O 接口芯片一个，它有两个 8 位的可编程的 I/O 口，全供用户使用。

7. Z80—CTC 计数器/定时器芯片一个，它有四个通道：通道 0 供用户使用，其余

由 CJBUG 使用。

8．按键共 28 个，16 个为十六进制数字键；12 个为命令键，具有二十种功能

MON′（换上档的监控）键

MON（换下档的监控）键

STEP（单步执行程序）键

EXEC（连续执行程序）键

TROM/MEM（EPROM 预检功能/存储单元检查）键

REG′/REG（辅助寄存器检查/寄存器检查）键

2FBE/LAST（用户定义功能/查上一个存储单元）键

2FBC/NEXT（用户定义功能/增量检查）键

PROM/SI（EPROM 写入/单指令调试）键

DELETE/INSERT（删除指令/插入指令）键

DUMP/PORT（信息转储磁带/口检查）键

LOAD/BP（磁带输入/设置断点）键

9．显示器有六位 LED 数字显示，通常左四位显示地址，右两位显示数据。

10．配有音频盒式磁带机接口。

11．布线区为 2.5×7 英寸。

12．S—100 总线插脚。

13．电源为 +5V±5%，1A。本机可以增配 7805 稳压器件，这时输入须接≥+7V 直流电源。若要对 EPROM 进行写入，尚须接入 +25V±1V，30mA 的电源。

1.3 功能简介

CJ801—Z80 单板计算机系使用 Z80 系列器件。Z80—CPU 的指令系统有指令 158 条，其中包含了 8080A 的全部指令。此外，还增加了数据块传送与查找、位操作、(置1、置 0、测试)、相对转移、变址寻址、16 位的算术运算等。Z80—CPU 的内部寄存器也较 8080A 多，增设了 IX、IY，以及一套辅助寄存器。这些特点大大增强了 Z80—CPU 的处理功能，并非常便于程序设计 Z80 的软件与 8080A 的软件相容。

本机使用八个廉价的静态读写存储器 2114，容量为 4K 字节。板上配有 2K 字节的 ROM，存放监控程序 CJBUG，便于用户利用键盘或盒式磁带输入程序和数据，以及提供其他的软件功能。板上设有一个完整的 EPROM 编程器，允许用户将信息写入 2716/2758EPROM 中。编程所需的电压为直流 +25V±1V，最大电流为 30mA。

本机配有音频盒式磁带机接口。用转录线将盒式磁带录音机与机上插孔相连，可以方便地存取数据和程序。单板机上的红色发光二极管能指示磁带上有无信息。利用这只二极管的指示，可在一盘磁带上录制几个文件。RAM 和磁带之间的存取由命令键控制。板上配有六位数码显示，供显示地址和数据。通常左四位显示地址，右二位显示数据（对于 PC，SP，IX，IY，等16位寄存器而言，右四位显示数据）。板上还配有 16 个十六进制的数字键和 12 个命令键，它们的功能在第二章里介绍。

本机配有三个按钮和开关：按钮 S1 用来对整机提供 RESET(复位)信号。开关 S2 有

两个位置可供选择，如果置向 MONRST 位置，则在复位后，显示器上出现“-”，并扫描键盘的输入；如果指向 PROM1 RST 位置，则在初始化后进入 PROM1 中的用户程序。开关 S3 用来选择 PROM2 中的 EPROM 处于“写入 (PGM)”或“读出 (READ)”的状态。

此外，板的左上方有 S—100 总线的插脚。Z80—CPU 的地址、数据、控制总线都接到S—100总线，以便在插座上扩充存储器板及 I/O 接口板。板的左方有 2.5×7 英寸的布线区、约可安装 25 片 IC (集成电路)。连接到布线区的信号有：

Z80—CPU 的数据、地址和控制总线。

Z80—PIO 两个口(Port)的数据线和联络线。

Z80—CTC 通道 0 的输入和输出。

1.4 操作步骤及注意事项

一、请按以下规定连线：

1. 直流稳压电源、盒式录音机均接至交流 220V。

2. 用转录线将单板机上的 AUX 插孔与录音机上的 MIC 插孔相连；

3. 用转录线将单板机上的 EAR 插孔与录音机上的 MONITOR OUT 或 EARPHO-NE 插孔相连；

4. 将单板机上的电源线接入直流稳压电源的 +5V 输出。

二、设定单板机的开关位置：

1. 通常将开关 S2 设定在 MONRST 位置；

2. 通常将开关 S3 设定在 READ 位置。

三、开机：

1. 接通直流稳压电源上的 +5V 开关；

2. 按下 RESET 按钮 S1，计算机进入监控程序 CJBUG，显示器上显示“—”，表明机器可接收命令。

四、输入用户程序，并进行相应的调试和操作。请参阅第二、四章。

五、如果进行第三步时，不出现“—”，则须立即关闭电源，仔细检查导线连接有否出错，器件与插座的接触是否可靠，各开关位置的设定是否正确，直流稳压电源的输出电压是否满足 +5V±5%。

六、切忌用手直接触摸集成电路，切忌对本机直接使用电源电压为交流 220V 的电烙铁，以防损坏芯片。必须使用电烙铁时，应将电源断开，利用烙铁的余热进行焊接。

1.5 《Z80 袖珍设计手册》使用说明

CJ801—Z80 单板计算机提供用户一本《Z80 袖珍设计手册》。这本小册子是扼要叙述 Z80—CPU 指令系统及其他芯片的编程要点。

本册第一页是 Z80—CPU 内可供用户使用的寄存器。其中有一套与 8080A 兼容的主寄存器，另有一套相同的辅助寄存器。此外还有一套专用寄存器。它们的使用请参考《微

处理器实用教材—基础知识和Z80器件的使用》。第2、3页是摘要表示指令是如何影响各个标志位的，第4、5页是8位传送指令组，黑体字的指令操作码与8080A相兼容，其余的是Z80所独有的新指令。新指令是变址的传送指令。第5页是对这些指令进行更为详细的说明。第6、7页是16位传送指令组，BC和DE的内容可以直接由存储单元输入，而不必通过HL寄存器。第8、9页是交换指令组和数据块传送和查找指令组。第10、11页是8位算术和逻辑指令组，其中大多数与8080A相兼容，新增加的指令也是一些变址操作指令。第12、13页是其他指令组，其中有求补码指令和中断方式选择指令。中断方式0是仿照8080A的，而方式2为Z80系列接口芯片所使用。

第14、15页是16位算术指令组。除增加了变址操作外，还增加了两种只有在小型计算机才具备的ADC和SBC指令。第16、17页提供了远较8080A更丰富的移位和循环移位操作的指令组。这些操作还能在以HL，IX和IY作为指针的存储单元上进行，就象6800那样。此外还能将存储单元中的BCD数字和累加器A进行移位操作。第18、19页是位操作指令。每一位都可被置位、复位和测试。第20、21页是转移指令组。其中有二字节的相对转移，而8080A只有三字节的转移指令。第22、23页是子程序调用和返回指令组，其中只增加了两条新指令，RETI和RETN，这两条是从中断和不可屏蔽中断返回的指令。第24、25页是输入/输出指令，新增加的指令为以C寄存器作为I/O的指针，数据传送的对象可以是CPU内8位寄存器。此外还可以进行数据块的输入/输出。第26页是Z80—CPU的中断结构。第27、28页是Z80—PIO和CTC的编程要点。第29、30、31页是Z80—SIO编程要点。最后是ASCII字符表。

第二章　键盘操作说明

在这一章，逐个介绍本机中各个按键的作用，并扼要介绍监控程序 CJBUG。更为详细的说明请参阅第四章及附录三。

2.1　监控程序 CJBUG 简介

CJBUG 固化在可擦的可编程序只读存储器(EPROM)中，在存储空间中它占用的地址为 $0000—07FF_H$，此外，CJBUG 还使用了 112 个 RAM 单元。有了 CJBUG，就可以通过键盘向计算机输入机器语言级的程序数据。

CJBUG 可以划分为下列四个主要程序段：(1) 初始化、显示和键盘分析程序；(2) 键盘动作程序；(3) 实用子程序；(4) 表格。

一、初始化、显示和键盘分析程序

1．初始化($0000—00EA_H$)

它设定栈指针 $SP=2FBA_H$，并将其保存在 $2FD9—2FDA_H$ 单元，然后设定 CJBUG 的栈指针 $SP=2FA2_H$。清除键盘状态标志。在最左端显示器对应的显示单元 DISEMEM 中填入"—"的代码 11H，其余填入空格的代码 10_H。

2．显示 DISUP($00EB—00ED_H$)

它将六个用于显示的显示缓冲单元(DISMEM—DSMEM5)的内容依次轮流取出，送往 LED 显示器，每位数字保持显示一毫秒。

3．键盘分析($00EE—01A2_H$)

它等待并搜索键盘的输入。若没有按键按下，则又回到显示 DISUP 程序。若有按键按下，则对所按下的键进行分析。如果是数字键，则送入相应的显示单元；如果是命令键，则进入与该键功能相对应的键盘动作程序。

二、键盘动作程序($01E4—0564_H$)

本机有十二个命令键，有二十个按键动作程序。这些动作程序的内容将在后面关于键盘操作的叙述中作简要介绍。

三、实用子程序($0565—07A5_H$)

这些子程序都是供 CJBUG 中上述程序段使用的，也可以由用户调用。其中也包括一些中断服务程序。

下面介绍几种常用的子程序。

1．UIX3（起始地址为 0659_H）

它使IX寄存器增3，B寄存器减1。该子程序只影响IX，B，F三个寄存器。

2．UFOR1（起始地址为 0679_H）

它将累加器A中的高4位作为一个十六进制数字写入(IX)单元，将低4位写入(IX+1)单元。该程序影响的寄存器有：A，B，E。

3．D20MS（起始地址为 0661_H）

它仅起延时20ms的作用，即调用这个子程序后，经过20ms后才返回。该程序影响的寄存器有H，L，E。

4．UABIN（起始地址为 $06BB_H$）

它将累加器A中的一个ASCII字符转换为二进制数，再送回累加器A。该程序使用的寄存器有A和F。

5．UBASC（起始地址为 $06C3_H$）

它将累加器A中的低4位作为一个十六进制数字，转换为ASCII字符，再送回累加器A。该程序使用的寄存器有A和F。

6．DISUP（起始地址为 $057C_H$）

功能，将存放在显示缓冲单元（DISMEM—DSMEM5）的内容依次轮流取出，送往LED显示器，每位数字保持显示一毫秒。

四、表格（$07A6—07FF_H$）

2.2 复位（RESET）按钮

本机中有两种实现复位的方法：一种是上电复位，即一合上电源即自动地提供复位信号；另一种是压下印刷线路板左右方按钮开关S1。

当开关S3处于READ位置，S2处于MON RST位置时，复位按钮的作用如下：

1．使Z80—CPU处于初始状态。这些初始状态包括：

1）中断允许触发器处于禁止状态；

2）置寄存器I，R的内容为 00_H；

3）置程序计数器PC的内容为 0000_H；

4）置中断方式为IM0。

在复位信号有效期间，地址总线和数据总线处于浮动状态，所有的输出信号均为无效。

2．使本机从 0000_H 开始执行监控程序的初始化部分，详细内容请参阅上节及附录三。此时，显示器的最左端应显示CJBUG的标志符号"—"。

3．如果用户程序中用BREAKPOINT键设置了断点，则压下复位按钮后所设置的断点数标志单元被清除。

4．如果开关S2处于PROM1 RST位置，则CJBUG执行完初始化程序后，自动转移到PROM1中起始地址为 0800_H 的用户程序。这个功能使用户能方便地转入PROM1中的用户程序。而不必从键盘输入命令。

2.3 16个十进制数字键

这些键用来向计算机输入十六进制数字。这些数字可以是存储单元的地址、I/O 口地址、寄存器标号、指令码，也可以是一些数据。每压下一个数字键，此数字即存入相应的显示缓冲单元。最先输入的数字送(DISMEM = $2FF7_H$)单元，依次为(DSME1 = $2FF8_H$)，……(DSMEM5 = $2FFC_H$)，(DSMEM6 = $2FFD_H$)，(DSMEM7 = $2FFE_H$)。并将前六个单元的内容显示于六位 LED 显示器上，(DISMEM)单元在最左边的 LED上显示出来。具体操作如表 2.1 所示。

数字键与命令键的联合使用见以下各节。

表 2.1　　数字键使用举例

按键	显示		说明
MON	—		压下 MON 键或按复位按钮，显示"—"
2	2		压下第一个数字键 2
0	2 0		再压下数字键 0
3 4	2 0 3 4		先后再压下数字键3，4
B 6	2 0 3 4	B 6	先后压下数字键 B，6
7	2 0 3 4	B 6	压下第七个数字键 7 没有任何反应
8	—		压下第八个数字键，CJBUG 进行处理，由于没于命令键输入，连续输入 8 个数字毫无意义，所以控制进入 CJBUG 的初始化部分

2.4 新增加的命令键

CJ801 单板计算机配用的新管理程序，CJBUG 仍为 2K 字节，在保持原监控程序(TP—BUG)的全部功能前提下，采用一键两用的双功能键，新增了九个命令功能。它们是插入功能(INSERT)删除功能(DELETE)、减量检查功能(LAST)，单指令执行功能(SI)、EPROM 预检功能(TROM)、内存清零功能(CRAM)，内存检查功能(TRAM)，上下档命令选择功能(MON′)，及用户自定义键功能。以外，还对断点设置命令(BP)和 EPROM 写入命令(PROM)进行了修改。

CJ—801 单板计算机配用的键盘如图所示：

PROM / SI	DELETE / INSERT	DUMP / PORT	LOAD / BP	
TROM / MEM	REG′ / REG	2FBE / LAST	2FBC / NEXT	
7 / H	8 / L	9	A	MON′
4 / IX	5 / IY	6 / I	B	MON
1 / PC	2 / SP	3 / IFF	C	SS
0	F	E	D	EXEC

2.5 MON(MONitor 监控)键

MON 键有两种功能。一是中止现行程序的执行。当计算机执行 RAM 中的用户程序时(也即用户已压下过 EXEC 键)，压下 MON 键，通过 U15 单稳电路向 CTC 通道 2 的 C/T2 输入一个脉冲。由于在 EXEC 键的键盘动作程序(起始地为 $02C7_H$)中，将 CTC 通道 2 设定为计数器工作方式，其初始常数为 01_H，因而，C/T2 的脉冲导致 ZC2 上产生一个脉冲。这个脉冲通过 U33 和 U34 送往 CPU 的 $\overline{NMI}$ 输入端，从而产生不可屏蔽中断，程序转入 0066_H 的中断服务程序。

这种功能在分析程序故障时尤其有用。例如，如果由于程序设计上的错误，或者由于执行了一条暂停(HALT)指令，或者由于陷入无休止循环，那么用户可以使用 MON 键，使控制返回到监控程序 CJBUG 而能保护 CPU 寄存器内容。这样，用户就能知道压下 MON 键时程序执行到何处。

MON 键的另一种功能是，中止或者退出当前的命令状态或输入数据，使控制返回到 CJBUG 的初始化部分，等待下一个按键输入，以便输入新的命令或数字。

通常要进行一种新的操作方式之前，均需压下 MON 键使显示器上出现"—"后，才能压下新的命令键或数字键。

MON 键与复位按钮的作用相似，都是使控制返回到 CJBUG。其不同点是：MON 键能保护 CPU 寄存器的状态以及 CJBUG 所使用的一些 RAM 专用单元(主要是键盘状态标志)。表 2.2 是说明 MON 键使用的例子。

表 2.2 MON 键使用举例

按键	显示		说明
MON	—		准备接受命令
2 0 0 0	2 0 0 0		输入四个数字，作为存储单元
TROM/MEM	2 0 0 0	××	地址检查2000单元，××为读出值
MON	—		退出以上工作方式
8 4	8 4		输入口地址
DUMP/PORT	8 4	××	检查口84，××为读出值

2.6 MEM(MEMory储存器检查)键

MEM键用来检查或更改RAM单元的内容，也就是通过键盘命令对RAM单元进行读、写操作。此外，也可对ROM或PROM进行读操作。

当显示器最左端"—"显示后，通过键盘输入表示地址的四个十六进制数字，先送地址的高位数字，后送低位数字，每送入一个数字，就立即显示在显示器上，最高位显示在最左端。然后压下MEM键，则在最右边两个显示器上出现该存储单元的内容，如果还没有送完四个数字的地址就压下MEM键，那么压下此键不起任何作用，右边两个显示器上不会出现数字，也不影响以前输入的数字。在这种情况下，如果继续输入数字，在送完地址的四个数字之后再一次压下MEM键，那么在显示器上仍能出现该单元的内容。

表 2.3 MEM 键使用举例

按键	显示		说明
2 1	2 1		压下两个数字键
TROM/MEM	2 1		此键不起任何作用
3 4	2 1 3 4		继续输入数字3、4
TROM/MEM	2 1 3 4	5 6	显示该单元内容5、6

其次，介绍如何更改存储单元的内容。(参阅表2.4)

在完成上述检查存储单元内容的操作之后，显示器上有六个数字，左边四个数字为地

址，右边两个数字为存储单元的内容。如果再输入两个数字，这两个数字即被写入该存储单元，并被显示出来，原先的存储单元内容随即消失。值得注意的是，只有送完数字后，显示才会更新。如果只输入一个数字，则这个数字保存在（DSMEM6 = $2FFD_H$）单元中，不起任何作用。只有当第二个数字输入后，才会将这两个数字写入该存储单元中，然后从此存储单元中送至DSMEM4和DSMEM5单元中，并被显示出来。如果由于偶然的疏忽，用户企图更改ROM或空单元的内容，虽然在DSMEM6和DSMEM7单元会保存新输入的数字，但是无法通过存储单元而进入DSMEM4和DSMEM5单元，从而不会更改右边显示的内容。此时用户会意识到自己的操作错误。

如果连续压下MEM EXAM键，则其结果只是重复读出并显示出该存储单元的地址和内容。如果压下MEXT键，则显示下一个单元的地址和内容。

2.7 PORT(PORT口 检查)键

PORT键的功能与MEM键相似，它用来检查或更改口的内容。

当显示器最左端出现"—"标志后，通过键盘输入口地址的两个十六进制数字，先送地址的高位数字，后送地址的低位数字。由于最多只能访问256个I/O口，所以口地址只需要两个十六进制数字。然后压下PORT EXAM键，则在最右边两个显示器上将出现该口的内容。由于它的功能与MEM EXAM键相似，请读者参阅上节和本节的例题，例中将扼要地介绍这些特性，从而不再作文字的叙述。

值得注意的是如果连续压下PORT EXAM键，则将连续显示该口的内容。这种特性对于CTC芯片是很有用的，通过这种操作可以了解CTC芯片中减1计数器的内容的变化情况。

压下MON键，可以使之退出PORT EXAM工作方式。

本机中共用11个I/O口，如表2.5所示。顺便指出，有些口只能读出，有些口只能写入，有些口（如PIO）包含若干个寄存器，读出和写入的操作与操作方式有关，使用时应予注意。

2.8 REG(REGister 寄存器检查)键

REG键用来检查或更改下列CPU寄存器的内容：A，B，C，D，E，F，H，L，I,IFF，PC，IX和IY。SP寄存器的内容只能被检查，不能被更改。此键的使用与MEM EXAM相似，唯一的不同是，它不能和NEXT键配合使用，因为CPU寄存器没有地址。

先压下代表寄存器号的数字键，在显示器的左边出现数字。然后压下REG EXAM键，显示器右边的两个或四个数字即为寄存器中的内容。IFF是中断允许触发器的状态，00表示禁止中断，04表示允许中断。

如果要改变所显示寄存器的内容，只需再压下两个数字键(若是IX、IY或PC，则需要输入四个数字)。

表 2.4　用 MEM 键更改存储单元的内容

按　　键	显　　示		说　　明
MON	—		或按复位按钮
2　0　0　0	2000		依次输入存储单元地址
TROM/MEM	2000	××	××为原读出值
0　6	2000	06	用 06 更改原来内容
4	2000	06	只输入一个数字，不会更改原来内容
5	2000	45	送完两个数后，更改原来内容
2FBC/NEXT	2001	××	显示下一个单元的地址和内容
7　B	2001	7B	用 7B 更改原来内容
MON	—		准备检查新单元0000H
0　0　0　6	0006		地址号错了，应是0000
MON	—		消去上述数字
0　0　0　0	0000		输入正确的地址号
TROM/MEM	0000	31	读出 0000H 单元内容

表 2.5　CJ801 使用的 I/O 口

口　地　址	口	说　　明
80H	PIO A 口数据寄存器	口内有数据输入寄存器和数据输出寄存器，读写过程与操作方式有关
81H	PIO B 口数据寄存器	同　上
82H	PIO A 口控制寄存器	只能写入，不能读出
83H	PIO B 口控制寄存器	同　上
84H	CTC 通道 0	可读写，但读出的是减计数器内容
85H	CTC 通道 1	同　上
86H	CTC 通道 2	同　上
87H	CTC 通道 3	同　上
88—8BH	七段选择锁存器 SEGLH	只能写入，不能读出
8C—8FH	数字选择锁存器 DIGLH	同　上
90—93H	键盘选择三态输入缓冲器 KBSEL	只能读出，不能写入

表 2.6　PORT 键使用举例

按　键	显　示		说　明
MON	—		准备接受命令
8	8		输入口地址
DUMP/PORT	8		未输完口地址即压下 PORT 键，不起任何作用
0	8 0		输完口地址80
DUMP/PORT	8 0	××	查口的内容，××为读出值
2FBC/NEXT	8 1	××	查下一个口的内容，××为读出值
MON	—		准备接受命令，检查口90
8 8	8 8		输入错误的口地址88，应为90
MON	—		准备接受新的命令
9 0	9 0		输入正确的口地址90
DUMP/PORT	9 0	7 F	检查口 90 的内容

用户可以压下 MON 键使之退出 REGEXAM 工作方式，或检查另一个寄存器。

在 CJBUG 的大多数按键工作方式下，CPU 寄存器的内容都保存在 RAM 的“用户寄存器存放区”。每当压下 EXEC 键(连续执行程序)或 SINGLE STEP 键(单步执行程序)而进入这两个按键的键盘动作程序时，此动作程序首先将“用户寄存器存放区”的内容送入各 CPU 寄存器而后进入用户程序。每当执行程序时遇到了断点或者单步执行程序结束时，或者用MON键中止用户程序的执行时，也就是说，每当退出用户程序进而入CJBUG的控制时。CPU 中各个寄存器的内容又推入“用户寄存器存放区”，以便用户进行更改和读出。

2.9　REG′ (alternate REGister辅助寄存器检查键)

REG′ 键用来检查或更改下列CPU辅助寄存器的内容：A′，B′，C′，D′，E′，F′，H′ 和 L′。用“′”来表示辅助寄存器。该键的操作和使用与 REG 相同，不再赘述。

表 2.7 REG键使用举例

按键	显示		说明
MON	—		准备接受命令
A REG′/REG	A	××	检查累加器A的内容，××为读出值
8	A	××	用8 4写入A，输入第一个数字8没有反应
4	A	8 4	两个数字8 4都输入后，A内容被更改
MON	—		准备接受命令
2 REG′/REG	2 2F	B A	检查SP的内容，为$2FBA_H$
2 F C 2	—		企图更改SP的内容为2FC2，没有成功
1 REG′/REG	1 20	0 0	检查PC的内容，为2000_H
2	1 20	2 0	2进入显示
3	1 20	2 3	3进入显示
4	1 20	2 3	没有响应
5	1 23	4 5	用2345更改PC的内容
2FBC/NEXT	—		NEXT键不起作用

2.10 BP(设置断点)键

该键用来在用户程序中设置一至五个断点。在调试用户程序时，利用断点可以在用户程序的某处暂停执行，以便用户检查上一段程序设计中有否发生差错，进而修改程序或者修改寄存器中I/O口中的内容。

设置断点的操作如表2.8所示。每当输入一个断点后，断点标志(BFLG＝$2FDD_H$单元)内容增1，断点地址被送往起始地址为$2FDE_H$的"断点表"BP。该表共有15个单元，每个断点占用三个存储单元，其中两个存放断点地址，另一个存放用户程序断点处指令操作码的第一个字节。在输入断点地址后按下BP键，显示地址应暗后复明，表示断点已被接收。如果输入断点多于五个则多设的断点输入将是无效，此时显示器上出现标志"—"。最初输入的五个断点仍保持不变。

表 2.8 设置断点的操作举例

按键	显示		说明
MON	—		准备接受命令
2 0 1 5	2 0 1 5		先输入第一个断点地址
LOAD/BP	2 0 1 5		断点被接受，(BFLG)=1,2025被送往(2FDE)(2FDF)单元
MON	—		
2 0 2 4	2 0 2 4		输入第二个断点地址
LOAD/BP	2 0 2 4		第二个断点被接受，(BFLG)=2，2024被送往(2FE1)(2FE2)单元
MON	—		
2 0 0 0	2 0 0 0		输入要执行的程序的起始地址
EXEC	2 0 1 5	××	从 2000_H 单元开始执行程序，暂停于第一个断点处显示断点地址(即PC)及累加器A的内容
EXEC	2 0 2 4	××	继续从 2015_H 单元执行程序暂停于第二个断点处，显示该断点地址(PC)及累加器A的内容，上述三步可反复进行
MON	—		控制转回 CJBUG
LOAD/BP	—		由于没有送完四个数就压下BREAK POINT键，从而使(BFLG)=0

当执行设有断点的用户程序时，先执行第一条指令，然后进入不可屏蔽的中断服务程序，装配断点，即将 RST8 (CF_H) 单字节指令代替各个断点的指令操作码的第一个字节，而此指令操作码的第一字节均被送往 BPTAB，然后继续执行程序，直到断点处执行 RST8 指令。接着进入起始地址为 0008_H 的服务程序，它暂停程序的执行，CPU 寄存器的内容均保存在“用户寄存器存放区”并将断点表 BPTAB 中所有的断点指令操作码的第一个字节(不管是否已经执行)送回用户程序，取代 RST8 指令，以便用户进行查检或更改，此时显示 PC 和 A 的内容。

如果按下 EXEC 键，可继续执行用户程序；如果程序中还有其余断点，则其过程与上述相同。

在每个断点处暂停用户程序时，用户可用 MON 等键对已执行的程序和执行结果(寄存器和 I/O 口)进行检查或修改。

如果用户要清除已设置的断点，可以使用下列方法：

在输入断点地址的四个数字之前，也就是说在输入零到三个数字后，紧接着压下 BP

键。从而将断点标志(BFLG)清零，并恢复用户程序的操作码。

通常MON键对设置断点没有任何影响，但是下列情况是例外。如果程序遇到 HALT 指令而停止，此时压下 MON 键，使控制转回 CJBUG，但是用户程序中的 RST8 指令并没有被取代而仍然留在用户程序中。在调试过程中，用户应该注意这种情况的发生。

2.11 SI(单步执行程序)键

此键可以用来每次执行程序中的一指令。执行完后，显示 PC (即下一条指令的地址) 和 A 的内容。此时，用户可以使用 MON 键以及其他的按键来检查或修改程序、I/O 口或 CPU 寄存器。

用户可以反复地压下 SI 键，使程序一步一步地执行，显示器上可以看到下一次要执行的指令的地址。对于条件转移(condional jump)和条件转子(conditional call) 指令，这种特性是很有用的。用户可以了解程序是否发生转移以及跳转到什么地址，也就是说，测试的"条件"是否满足。在一步一步地执行程序时，用户可以更改 PC 的内容，强使程序转移到一个新的地址。

不仅 RAM 中的程序可以单步地执行，而且 ROM，PROM1，EPROM 中的程序可以单步地执行。值得注意的是，单步执行与 CTC 通道 2 有关的指令，很容易产生差错，用户应尽量避免这种情况。

2.12 EXEC(EXECute 连续执行程序)键

这个键用来连续执行 RAM，ROM 或 EPROM 的程序。它有两种使用方式：

1. 先输入要执行的程序起始地址的四个数字，然后压下 EXEC 键，即从地址开始执行程序。如果要从程序的起始开始执行，通常使用这种操作方式。见表 2.8。

2. 仅仅压下 EXEC 键。此时从保存在"用户寄存器存放区"中的现行地址开始执行程序，如果在连续执行程序时遇到断点，或是单步执行程序而暂停程序的执行，而此后的要求连续执行程序，或只需压下 EXEC 键即可。这种操用方式请参阅表 2.8，

2.13 DUMP(cassetteDUMP 信息转储)键

此键用来将 RAM 中的信息转储到盒式录音机的磁带中。在转储过程中使用美国"堪萨斯(Kansas)城标准"。传送信息的速率为 300 波特 (Baud)，即以八个 2400 赫音频脉冲信号表示"1"以四个 1200 赫音频脉冲信号表示"0"。在一个文件的转储过程中，开头有 40 秒全"1"的导引信号，末尾有 5 秒全"1"的结尾信号。

转储的操作过程如下：

1. 用转录线将本机 J2(AUX)端与录音机的"AUXIARY"或"MIC"输入端相连。

2. 将磁带装入录音机。

3. 利用 MEM EXAM键，将内存中需要转储的信息的起始地址置入 $2FC0_H$ 和 $2FC1_H$ 单元(高字节置入 $2FC0_H$，低字节置入 $2FC1_H$)。

4．利用 MEM EXAM键，将内存中需要转储的信息的最终地址置入 $2FC2_H$ 和 $2FC3_H$ 单元(高字节置入 $2FC2_H$，低字节置入 $2FC3_H$)。

5．压下 MON'键，使显示器出现"，"。将录音机置成录音方式，然后压下 DUMP键。"，"将消失。

6．不需要进行音量调节，因此录音机内有 AGG 自动增益控制。当转储完成后，显示器上再次出现标志"，"。此时按下录音机的 STOP 键，停止走带。整个转储过程至少要 45 秒。

2.14 LOAD(cassette LOAD 磁带输入)键

此键将用来录音机磁带中的信息输入 RAM。输入的操作过程如下：

1．用转录线将录音机的"MONITOR OUT"或 EARPHONE"孔与本机的 JI(EAR)孔相接。

2．将磁带走到相应位置。

3．将录音机的高音制调到很大，低音控制到最小，音量控制到最小。

4．压下 MON'键，显示器上出现标志"，"。然后压下 LOAD 键，"，"标志消失。

5．将录音机置成放音方式，逐渐增大音量，直至 LED (发光二极管)出现亮光。然后再增大音量 20%。在整个输入过程中，LED 将保持发亮。

6．如果输入过程获得成功(即核实了各个记录的"检查和"Checksum*)，则 CJBUG 将显示标志"，"。此时可以关闭录音机。

7．如果在输入过程中发现某个记录的"检查和"有差错，则显示下一个要输入的记录的第一个字的存放地址。这种情况表明，刚才输入的记录有差错，而在这个有差错记录以前的输入是成功的。此时应重新进行输入，并核实音量和音调的设置是否恰当。

8．在同一条磁带上可以记录好几个文件(file)。当走带进入一个文件时 LED 显示器发亮；而进入文件之间的间隙时，则 LED 不发亮。用户可以利用这个特点来识别不同的文件。

用户也可以使用语言标志或录音机上的走带计数器对文件进行识别。

2.15 PROM(PROM，EPROM写入)键

在本机中，可以将RAM或ROM中的数组写入插在 PROM2 插座(起始地址为 1000H)中的 2716/2758 型 EPROM。对 EPROM 进行写入，还需要有一个 +25±1V，30mA 的辅助电源。此电源应接到印刷线路板上右边标有标记 +25V 的焊点上。

对 EPROM 进行写入的操作过程如下：

1．将需要写入的 EPROM 用紫外灯穿过器件窗口进行照射，以擦除其中原有的内容，即各单元内容均应为 FF。如果是未使用过的新器件，则可以免去此步。

---

* 常通要输入文件(可以是一批数据，也可以是一个程序)由好几个记录构成，记录的格式详见第四章，每个记录有一个"检查和"

所用紫外线的波长为2537Å，照射强度为20000μW/cm²，照射能量为15W—sec/cm²。通常可使用医用的紫外线灯(15～30W)，照射距离为2.5～5.0cm，照射时间为20—40分钟，即可擦除EPROM的内容。

2．在关闭电源的情况下，将EPROM插入插座。

3．合上+5V和+25V电源。

4．将目标数据首址送入$2FC0_H$ $2FC1_H$，将源数据首址送入$2FC2_H$、$2FC3_H$，先送高字节，再送低字节。

5．压下MON′键LED显示器上出现标志“′”。

6．向微型机输入四个十六进制数字，表示要向EPROM写入的字节数，先送高位数字。

7．将开关S3置于PGM位置，压下PROM/SI键，显示器将熄灭，并进行写入。每个字节的写入需要约52ms。

写入完毕后，有两种可能的结果：

一(多半)是显示器上出现标志“—”。这表示写入EPROM的内容与核对后无误。

另一是在显示器上出现四个数字，表示EPROM中第一个与源程序的内容不相等的单元的地址(左边四个数字)和内容(右边两个数字)。如果压下NEXT键，键盘动作程序将继续核对EPROM中的内容。如果没有错误，则显示器上出现“—”。如果有错误，则继续给予显示，直至用NEXT键检查完毕为止。在写入EPROM过程中出现错误，往往是由于使用了没有“擦净”的EPROM，或者所用的EPROM是废品。

在写入过程完成以后，将开关S3置于READ位置。

还有一点需要提醒用户注意，在EPROM写入过程中要插入52.5ms的等待(Wait)状态。在等待状态中Z80—CPU暂停对动态存储器的刷新，从而使动态存储器中的内容丢失。如果用户需要扩充使用动态存储器，则应对此问题予以注意。本机中使用静态存储器，故不会产生上述问题。

2.16 NEXT(增量检查)键

NEXT键用于下述三种操作方式：检查存储器，检查口，检查EPROM的下一个写入错误。它们的操作和特性已如前述，此处不再重复。

如果在其他操作方式中使用NEXT键，则不会产生任何反应。此时在显示器上出现“—”，控制转入CJBUG的初始化部分。

2.17 MON′(换上档的监控)键

此键除了具备MON键的基本功能——使系统进入监控程序的待命状态外，尚完成双功能键的上下挡更换。

例：TROM/MEM键，在按下MON′后再按此键。CJBUG接受上挡键命令，进行EPROM预检，操作参见表2.9。

2.18 TROM(EPROM预检)键

将待编程的 EPROM 插入规定的 $PROM_2$ 插座里,(内存地址为 1000_H),按下TROM、若显示"—"(提示符)表示芯片的内容为全1,若显示地址及数据,则表示这一单元不是全1。

新增的这一命令,可以对购置的 EPROM 一一进行简易检验。同时,也可用来检查 EPROM 经紫外线照射后,是否擦清。

例:固化于 EPROM 的程序经紫外线光照后,A 块全部擦清,B 块没有擦清,则按下 TROM 键后,状态如下所示:

表 2.9A　　TROM 键使用举例

按键	显示		说明
MON′	I		CJBUG 待命,上档命令有效
TROM/MEM	–		查 EPROM,内容为全1

表 2.9B　　TROM 键使用举例

按键	显示		说明
MON′	I		CJBUG 待命,上档命令有效
TROM/MEM	1000	AF	查 EPROM,1000 单元内容为 AF。

注:在进行此操作前,要在关闭电源的情况下,将 EPROM 插入 PROM2 座插。

2.19 INSERT(插入)键

在六位显示器左边四位显示一个地址,按一下 INSERT 键,则从所示的地址单元开始,直到 $2F01_H$ 单元为止,各存储单元内容均向下移动一个字节。然后用存贮器检查方法插入指令。

CJ801 单板机在其 4K 字节的RAM之中。分配给用户可用的区域为 2000_H 到 $2F89_H$。使用 INSERT 键可以移动的存储单元范围为 2000_H 到 $2F01_H$。也就是说,从 $2F02_H$ 至 $2F89_H$ 为止的 136 个字节不会向下移动,这 136 个字节可放置数据表格和变量。

例:表 2.10 显示 INSERT 键功能,所示存储单元内容是假设的,此操作的目的是想在 2300_H 处插入一条两字节的新指令。

表 2.10　　INSERT 键使用举例

	按INSERT前	按 INSERT	再按一次INSERT
显　示	2300 [　]	2301 [AF]	2302 [AF]
操作结果	2300　AF	2300　AF	2300　AF
	2301　0 4	2201　AF↙	2321　AF
	2302　7 5	2302　0 4↙	2302　AF↙
	2303　2F	2303　7 5↙	2303　0 4↙
	⋮	⋮　↙ ↙	⋮　↙ ↙
	2F00　FF	2F00　××↙	2F00　××↙
	2F01　BB	2F01　FF↙	2F01　××↙
	2F02　AA	2F02　AA	2F02　AA
	2F03　CC	2F03　CC	2F03　CC

• 注：在此操作后，用MON键退出命令，然后再用存储器检查方法插入指令。

2.20　DELETE(删除)键

先用MON′键，表示上档命令有效。然后，在六位显示器左边显示一个地址，按一下DELETE 键，则从所示的地址单元开始，直到2F01H 单元为止，各存储单元 内 容 均 向上移动一个字节，从而实现删除。

例：表 2.11 显示DELETE键功能，所示存储单元内容是假设的，此操作的目的是想

表 2.11　　DELETE 键使用举例

	按DELETE前 按MON′	再按地址	按DELETE	再按一次DELETE
显示	[ı] [　]	2300 [　]	2300 [01]	2300 [06]
操作结果	2300　3E	2300　3E	2300　01↖	2300　06↖
	2301　01	2301　01	2301　06↖	2301　08↖
	2302　06	2302　06	2302　08↖	2302　××↖
	2303　08	2303　08	2303　××↖	2303　××↖
	⋮	⋮	⋮　↖ ↖	⋮　↖ ↖
	2F00　78	2F00　78	2F00　00↖	2F00　00↖
	2F01　00	2F01　00	2F01　00	2F01　00
	2F02　11	2F02　11	2F02　11	2F02　11
	2F03　22	2F03　22	2F03　22	2F03　22

在 2300_H 处删除一条两字节的指令。

2.21 LAST(查上一个存储单元)键

这是新增加的一个常用命令键，用以访问显示器现行存储单元的上一个存储单元。这一点与 NEXT 键正好相反。但 NEXT 键不仅可以访问(读或写)下一个存储器单元，而且还可以访问下一个输入/输出口，以及 EPROM编程时的下一个错误检查。而 LAST 键只能检查存储器。这样，交叉使用 LAST 和 NEXT 键便能灵活地更换所要访问的存储单元。

例：假定 2000_H 至 2003_H 诸单元中存有 00、11、22、33 等数据，用 LAST 键检查他们，其操作如表 2.12。

表 2.12　　LAST 键使用举例

按键	显示		说明
MON	-		CJBUG 待命，下档键有效
2 0 0 3	2003		给出存储单元地址
TROM/MEM	2003	33	下档有效，存储器检查
2FBE/LAST	2002	22	下档有效，查上一个存储单元
2FBE/LAST	2001	11	查上一个存储单元
2FBE/LAST	2000	00	查上一个存储单元
2FBC/NEXT	2001	11	返回2001
A 6	2001	A6	写入 A6

2.22 SI(单指令调试)键

每按一次 SI 键，执行程序中的一条指令。执行完后，显示 PC (即下一条指令地址)和 A 的内容。此时，用户可以使用 MON 键以及其他的按键来检查和修改程序，I/O 或 CPU 寄存器。

此键与 SS (单步执行程序)键功能相类似，但是 SI 键对于 CALL 指令，也能一次执

行，而SS则不同。因此，在调试程序中，可以交叉使用SS和SI键，，从而提高调试的效率。

例：源程序是 2000 LD A 01　　　2200 LD B、A
2002 CALL 2200　　　2201 ADD A、B
2005 LD B、A　　　2202 RET
⋮　⋮
2029 HALT

要进行单指，单步调试，则可进行以下操作。

表 2.13　　　**SI键使用举例**

按　　键	显	示	说　　明
MON	-		CJBUG待命，下档键有效
1	1		PC寄存器访问，"1"代表"PC"
REG'/REG	1 ××	××	下档有效，"×"为无用数
2 0 0 0	1 20	00	输入源指令地址
SS	2002	01	单步执行命令，显示PC和A
PROM/SI	2005	02	单指令指行，显示PC和A

2.23 2FBC、2FBE(用户程序启动)键

这两个按键，留给用户定义，用于启动两个用户程序。在使用之前，应将用户程序的启动地址、置入用户程序启动地址表中($2FBC_H$～$2FBF_H$)。每个地址的低字节在前，高字节在后。操作参照表2.14。

2.24 DISP(相对转移偏移量计算)功能

此功能是进行相对转移指令操作数——偏移量的计算；显示计算结果；提示计算的有效性(是否超出单字节范围)；计算结果写入指令的操作数字节。与原监控相比，CJBUG选用寄存器对存储源指令地址，(IY寄存器对)和转移目的地址(IX寄存器对)，并且，程序的起始地址改为0508_H。

例：某JR指令的地址为2300_H，转移的目的地址为$22F0_H$，计算相对偏量的操作如表2.14所示。

表 2.14　　DISP 功能 2FBC 2FBE 使用举例

按　　键	显	示	说　　明
MON	—		CJ—BUG 待命，下档命令有效
2FBC	2FBC		访问 2FBC 存储单元
TROR/MEM	2FBC	XX	下档有效"X"为无用数
0 8	2FBC	0 8	输入 DISP 程序起始地址低字节
2FBC/NEXT	2FBD	XX	访问 2FBD 存储单元
0 5	2FBD	0 5	输入 DISP程序起始地址高字节
MON	—		退出当前命令
IY	5		IY 寄存器对访问，"5"代表 IY 键
REG′/REG	5　XX	XX	下档有效，"X"为无用数
2 3 0 0	5　2 3	0 0	输入源指令地址
MON	—		退出当前命令
IX	4		IX 寄存器对访问，"4"代表 IX
RFG′/REG	4　XX	XX	下档有效，"X"为无用数
2 2 F 0	4　2 2	F0	输入转移目的地址
MON′	′		退出当前命令，上档有效
2FBC/NEXT	FF	EE	计算结果为 EE。FF 为转负跳转

2.25 TRAM(内存检查)功能

将待检查的存储单元首址送 $2FC0_H$,$2FC1_H$,待检查的存储单元末址送 $2FC2_H$,$2FC3_H$;然后，键入内存检查程序的首址 $053F_H$，按下 EXEC（执行程序)键。若内存是好的,则显示提示符，操作如表 2.15 所示。若内存是坏的，则显示错误地址及内容，操作如 表 2.16 所示。

例：在内存检查前已经将 2000_H 置入 $2FC0_H$，$2FC1_H$，$27FF_H$ 置 入 $2FC2_H$，$2FC3_H$。在 2000_H 至 $27FF_H$ 内存是好的，则显示“—”。

表 2.15 TRAM 功能使用举例

按键	显示		说明
0 5 3 F	0 5 3 F		键入内存检查程序首址
EXEC	—		内存检查完，显示“-”

例：在内存检查前已 将 2800_H 置入 $2FC0_H$，$2FC1_H$，2BFF 置 入 $2FC2_H$ $2FC3_H$。如果 2800 单元的高四位出错，则显示其错误地址及内容，然后将 A 累加器内容与错误内容比较确定出错的2114芯片是高四位还是低四位。

表 2.16 TRAM 功能使用举例

按键	显示		说明
0 5 3 F	0 5 3 F		键入内存检查程序首址
EXEC	2 8 0 0	4 0	显示出错的内存单元及内容
MON	—		退出当前命令
A	A		正确内容放在 A 中，A 累加器访问
REG′/REG	A	0 0	与错误内容较，说明 D_6 出错

2.26 CRAM(内存清零)功能

将待清零的存储单元首址送 $2FC0_H$，$2FC1_H$，待清零 的 存 储 单 元 末 址 送 $2FC2_H$，

2FC3H，然后键入清零程序的首址 052BH，按下 EXEC 键(执行程序)即可。

例：将 2000H 至 27FFH 的内存清另，则操作如下：

表 2.17　　CRAM 功能使用举例

按　　键	显　　示		说　　明
MON	—		CJBUG 待命，下档命令有效
2 F C 0	2FC0		2FC0H 存储器访问
TROM/MEM	2FC0	XX	下档有效"X"为无用数
2 0	2FC0	2 0	输入待清零存储单元首址高字节
2FBC/NEXT	2FC1	XX	下一个存储器访问
0 0	2FC1	0 0	输入待清零存储单元首址低字节
2FBC/NEXT	2FC2	XX	下一个存储器访问
2 7	2FC2	2 7	输入待清零存储单元末址高字节
2FBC/NEXT	2FC3	XX	下一个存储器访问
F F	2FC3	FF	输入待清零存储单元末址低字节
MON	—		退出当前命令
0 5 2 B	0 5 2 B		输入清零程序首址
EXEC	—		将 2000H～27FFH 清零

第三章 CJ801-Z80 单板计算机的结构和原理

3.1 概 述

CJ801—Z80 单板计算机的原理框图如图 3.1 所示。它有三条总线：数据总线 DB、地址总线 AB、控制总线 CB。挂到这三条总线上的器件有下列三类：

1. Z80—CPU；
2. 存储器：有 ROM，PROM1，PROM2，RAM；
3. 接口电路：有Z80—PIO，Z80—CTC，键盘和显示，录音机的接口；

此外，尚有译码电路、时钟和复位电路等。详细的原理图请见附录一。

下面对电路的各个部分以及本机的一些功能作进一步的介绍。

3.2 时钟电路

时钟信号用来协调微型机内各部分的动作，使其有条不紊地进行操作。时钟电路示于

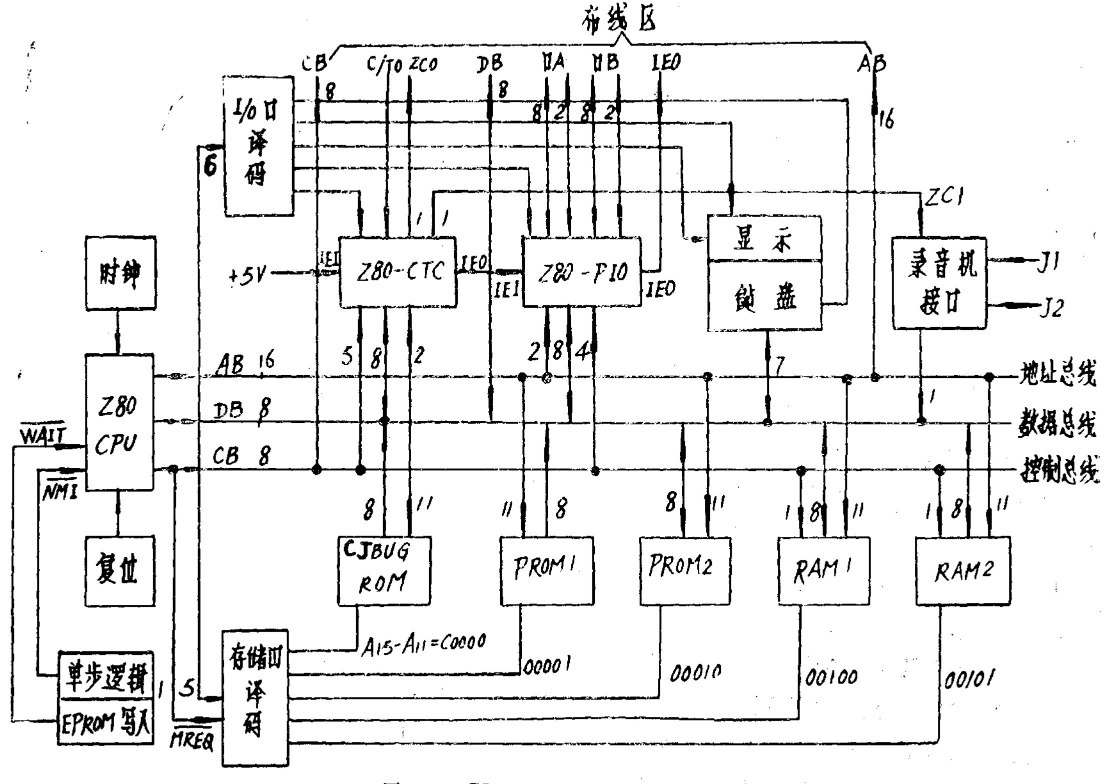


图 3.1 CJ801单板计算机原理框图

图 3.2。晶体振荡电路的振荡频率为 3.9936MHz(兆赫)，这个数值恰好是堪萨斯标准音频磁带机接口所要求的 1200/2400Hz(赫)和 300 波特(Baud)的整倍数。这个频率由 D 触发器 U30 分频而成为 1.9968MHz 的 CPU 时钟频率。CPU 时钟频率略低于 2MHz，时钟周期接近于 500ns，从而可以使用 8080A 的接口芯片以及廉价的存储器。这个 D 触发器的输出通过 74LS04，反相器接到 Z80—CPU，PIO 和 CTC。

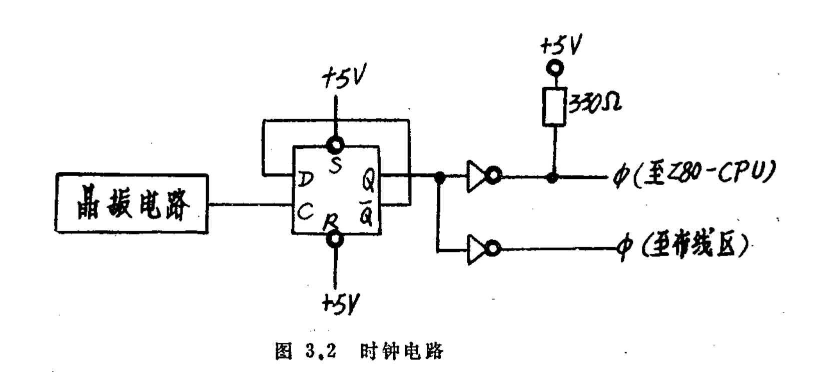


图 3.2 时钟电路

3.3 Z80—CPU

Z80—CPU 有一组 16 位的地址总线、一组 8 位的双向数据总线、一组 13 位的控制总线。13 个控制信号中本机使用 9 个，未使用 $\overline{HALT}$、$\overline{RFSH}$、$\overline{BUSRQ}$、$\overline{BUSAK}$ 信号($\overline{BUSRQ}$ 输入线接高电位，使其无效)。所有的总线均接向布线区和 S—100 总线插孔，以供用户附加电路时使用。关于 CPU 的详细介绍请参考《微处理器实用教材—基础知识和 Z80 器件的使用》及 Z80—CPU 技术手册。

3.4 存储器

一、存储空间分配

存储空间的分配及片选信号见表 3.1。

CJBUG 监控程序安排在最下面的 2K 字节(其地址号为 0000—07FF$_H$)。

0800—0FFF$_H$ 为 2K 字节的 PROM1，其中可存放用户的应用程序。

1000—17FF$_H$ 为 2K 字节的 PROM2，如果用户的应用程序超过 2K 字节，则可继续存放在 PROM2。此外，CJBUG 还可对该插座中的 EPROM 进行写入。

1800—1FFF$_H$ 的 2K 字节，没被使用。

2000—27FF$_H$ 为 2K 字节的 RAM1。

2800—2FFF$_H$ 为 2K 字节的 RAM2。

RAM(即 RAM1 和 RAM2)的存储分配如表 3.2 所示。

3000—37FF$_H$ 为 2K 字节，没被使用。

3800—3FFF$_H$ 为 2K 字节，没被使用。

表 3.1　　　　　　　　**CJ801 的存储分配**

地　　址	器　　件	A15-A11	A10-A0	译码器的有效输出
3800—3FFFH	没　用	00111	可　变	$\overline{Y7}=\overline{CS7}$
3000—37FFH	没　用	00110	可　变	$\overline{Y6}=\overline{CS6}$
2800—2FFFH	2K RAM2(U20-U23)	00101	可　变	$\overline{Y5}=\overline{CS5}=\overline{RAM2\ SEL}$
2000—27FFH	2K RAM1(U16-U19)	00100	可　变	$\overline{Y4}=\overline{CS4}=\overline{RAM1\ SEL}$
1800—1FFFH	没　用	00011	可　变	$\overline{Y3}=\overline{CS3}$
1000—17FFH	2K PROM2(U9)	00010	可　变	$\overline{Y2}=\overline{CS2}=\overline{PROM2\ SEL}$
0800—0FFFH	2K PROM1(U8)	00001	可　变	$\overline{Y1}=\overline{CS1}=\overline{PROM1\ SEL}$
0000—07FFH	2K ROM(U7)	00000	可　变	$\overline{Y0}=\overline{CS0}=\overline{MON\ SEL}$

表 3.2　　　　　　　　**RAM 的存储分配**

地址空间	用　　途	字节数	插座编号
2FFF 2FBA	CJBUG 使用的RAM暂存区和断点表	70	U_{20}—U_{23}　RAM_2
2FB9 2FA2	用户栈工作区	24	
2FA1 2F8A	CJBUG 栈工作区	24	
2F89 2F02	放置数据、表格、变量	136	
2F01 2C00	RAM_2 的用户工作区(1)	770	
2BFF 2800	RAM_2 的用户工作区(2)	1K	
27FF 2400	RAM_1 的用户工作区(2)	1K	U_{16}—U_{10}　RAM_1
23FF 2000	RAM_1 的用户工作区(1)	1K	

二、存储器译码

存储器的译码是由 U24 的 74LS138 八中取一译码器来完成的，具体电路如图 3.3 所示。译码器的每根输出接至 2K 存储器的 $\overline{CE}$ 端或 $\overline{CS}$ 端，故本机可配 16K 存储器。译码器规定 A15 和 A14 应为 0，故 16K 存储器的地址空间为 0000—$3FFF_H$，占有下方的 16K 字节。

译码器的输出都接向 U27 的 16 个引线孔，如图 I.1 所示。如果要改变各个 2K 存储器的地址空间，则可以切断印刷线路板引线孔之间的铜泊连线，按装 16 只脚的插座、其中插入具有相应连线的插头。

三、系统 RAM

U16—U19 为总计 2K 字节的 RAM(2114 型)，U20—U23 为另外的 2K 字节 RAM。RAM 中部分单元用作 CJBUG 的暂存区和栈区，如表 3.2 所示。

RAM 与 CPU 之间的连接如图3.3所示。

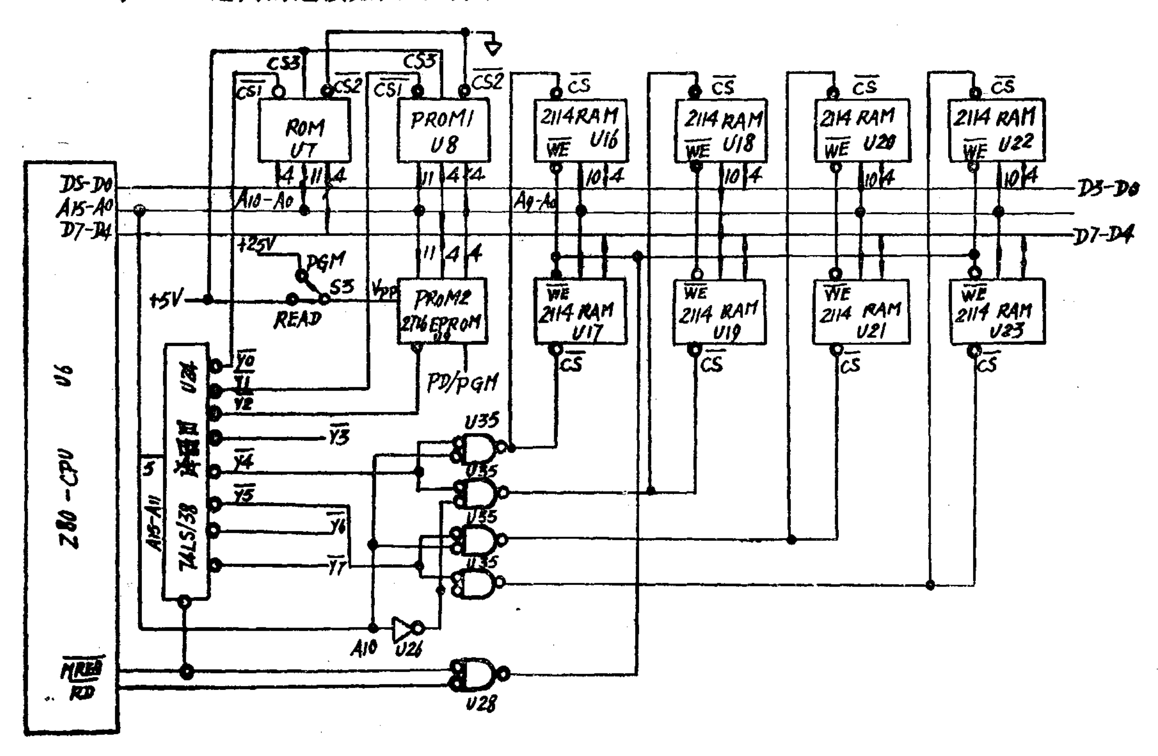


图 3.3 RAM与CPU之间的连接

四、PROM1

当 RESET 信号为有效时，从 CJBUG 的 0000_H 单元开始执行程序。在 $00E2_H$ 单元的指令检查开关 S2 的位置。如果 S2 置于 MON RST 位置，则继续执行 CJBUG 监控程序，在显示器上显示"—"，并扫描键盘的输入。如果 S2 置于 PROM1 RST 位置，则转移到起始地址为 0800_H 的 PROM1 中的程序，从而可以不必通过键盘输入命令而进入用户程序。

五、PROM2—EPROM

本节结合硬件和软件来解决 EPROM 的写入过程。

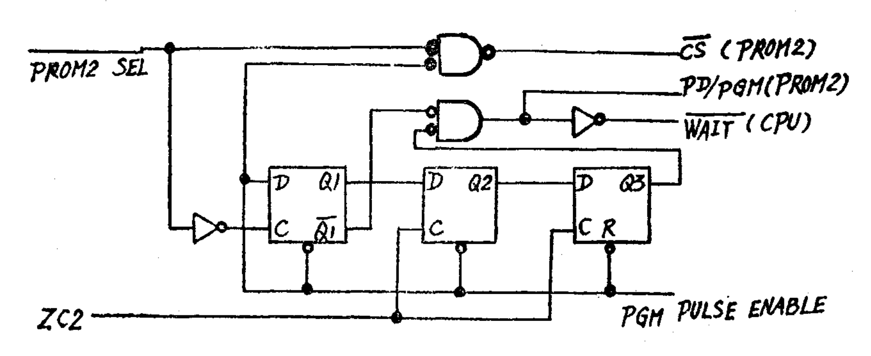
首先复习 2716/2758 型 EPROM 的操作方式，如表 3.3 所示。为了对 EPROM 进行写入，要求 S3 开关置于 PGM 位置，即 EPROM 的 Vpp 引脚接至 +25V 电源。此外要求 CS=1、PD/PGM 引脚上输入一个 TTL 电平的脉宽为 50—55 毫秒的正脉冲。在写入时，地址总线上应出现 EPROM 被写入单元的地址，数据总线应出现被写入的内容。

EPROM 写入的电路如图 3.4(a) 所示，操作的时间图如图 3.4(b) 所示。在写入前应做完一切准备工作，即用紫外线擦除 EPROM 中的内容即各单元内容都为全1。要写入

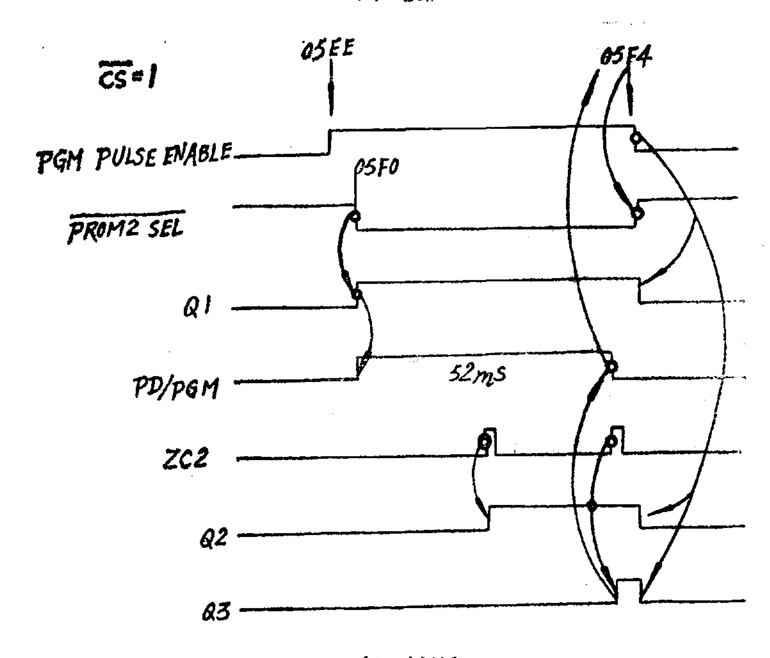
时，应在未接电源的情况下把 EPROM 片子插入插座 U9，再接上电源 +5V 和 +25V，将要写入的内容输入 RAM 单元，然后将该源程序的首地址按键输入到 2FC2，2FC3 单元，及把 EPROM 写入区的首地址输入到 $2FC0_H$、$2FC1_H$ 的单元中(先送地址的高位)，接着使 S_3 开关置于PGM位置，按下MON′键，将需要写入的字节数用数字键送入，字节数用四位十六

表 3.3　EPROM 的工作方法

工作方式＼管脚	PD/PGM	$\overline{CS}$	VPP	数据线状态
读	0	0	+5V	D_{out}
禁止	0	1	+5V	高阻抗
待机	1	×	+5V	高阻抗
写入	52ms 脉冲	1	+25V	D_{in}
校核	0	0	+25V	D_{out}
禁止写入	0	1	+25V	高阻抗



(a) 电路



(b) 时间图

图 3.4　EPROM 写入的电路和时间图

进制数表示，显示在左端四个 LED 显示器上，也就是说，它们存放在 DISMEM-DSMEM3 的存储单元内。最后压下 PROM/SI 键，进入该键的动作程序（起始地址为 $03C2_H$）：

首先将 CTC 通道 2 设定为定时器工作方式，每隔26毫秒址送出一个脉冲，但并不需要产生中断。其次执行 $03DC_H$ 单元的指令，使 PGM PULSE ENABLE 信号变高电平，从而撤除对 Q1—Q3 触发器的封锁。下一条为 03DE 单元的 LDI 指令，开始对 EPROM进行写入，例如地址总线上出现 1000_H，数据总线上出现 2000_H 单元的内容，$\overline{\text{PROMZSEL}}$变为低电平，Q1 变为高电平，PD/PGM 变为高电平，$\overline{\text{WAIT}}$ 变为低电平（有效），从而使 CPU 进入等待状态。当 CTC 通道 2 的 ZC2 发出第二个脉冲（即 52 毫秒以后）时，Q3 变高，从而使 PD/PGM 变低，$\overline{\text{WAIT}}$ 变高，CPU 结束等待状态。执行完 LDI 指令后，继续执行下一条指令（$03E2_H$），使 PGM PULSE ENABLE 变低，从而使 $\overline{\text{PROM2 SEL}}$ 变高、Q1—Q3 变低。这一单元的写入过程到此结束。写入下一单元的过程与此相同。

2758 与 2716 的写入过程基本相同，两者只有一只引脚不同，即 2758 的引脚 19 为 AR 而不是 2716 的 A10。通过跳接线（jumper）可将 AR 端接至 +5V，或接至地，或接至地址总线的 A10。

3.5 I/O 接口

一、I/O 译码及空间分配

I/O 译码电路如图 3.5 所示。U36 单元的八中取一译码器将 I/O 口每四个为一组，每次选中一组，如表 3.4 所示。表中 $\overline{\text{PS5}}$—$\overline{\text{PS7}}$ 连向布线区以供用户使用。对于 PIO 和 CTC，可以用 A1 和 A0 线将表中所示地址分配给有关单元。

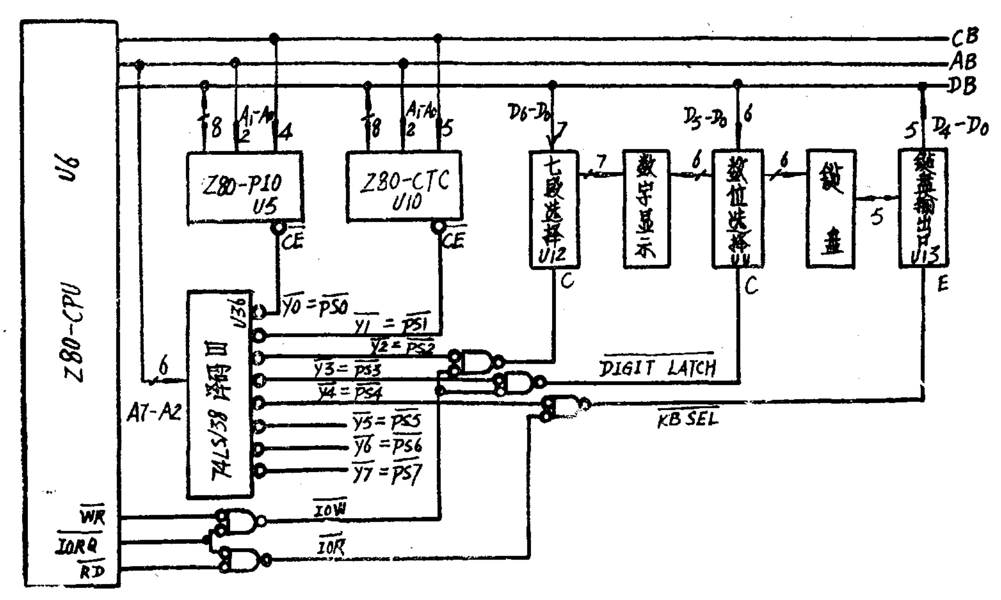


图 3.5 I/O 译码电路

二、Z80—PIO

Z80—PIO 是通用的并行 I/O 口芯片。它有两个 8 位的口。每个口有两根联络线。

表 3.4

A7—A2	译码器输出	器件	A1	A0	口	口地址
100000	$\overline{Y0}=\overline{PS0}=\overline{PIO\ SEL}$	Z80—PIO U5	0 0 1 1	0 1 0 1	口A数据寄存器 口B数据寄存器 口A控制寄存器 口B控制寄存器	80H 81H 82H 83H
100001	$\overline{Y1}=\overline{PS1}=\overline{CTC\ SEL}$	Z80—CTC U10	0 0 1 1	0 1 0 1	通道0 通道1 通道2 通道3	84H 85H 86H 87H
100010	$\overline{Y2}=\overline{PS2}=\overline{SEG\ LH}$	74LS273 U12 八锁存器	×	×	七段选择 (只写)	88—8BH
100011	$\overline{Y3}=\overline{PS3}=\overline{DIG\ LH}$	74LS273 U11 八锁存器	×	×	数位选择 (只写)	8C—8FH
100100	$\overline{Y4}=\overline{PS4}=\overline{KB\ SFL}$	74LS244 U13 八缓冲器	×	×	读键值 (只写)	90—93H
100101	$\overline{Y5}=\overline{PS5}$	没使用				94—97H
100110	$\overline{Y6}=\overline{PS6}$	没使用				98—9BH
100111	$\overline{Y7}=\overline{PS7}$	没使用				9C—9FH

Z80—PIO的两个口全部供用户使用，本机并不予占用，因而这些线都接向布线区，可供用户附加电路。

每个口都可以使用联络线进行中断控制，并采用IM2中断方式。

三、Z80—CTC

Z80—CTC有四个通道：

通道0—供用户使用，中断服务程序的起始地址为$2FD6_H$。用户可在$2FD6_H$处安排一个转移指令，转移到中断服务程序。

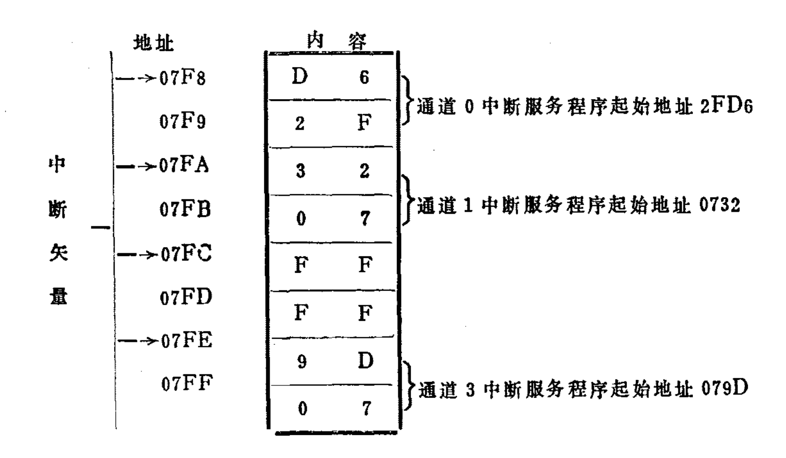
通道1—在本机中用于"DUMP"键的动作程序，该程序将RAM中信息转储到盒式录音机的磁带中。

通道2—在本机中用于PROM/SI键、SINGLE STEP键和MON键的动作程序，在PROM/SI键动作程序中，通道2仅用来产生52ms的脉冲。在SINGLE STEP键和MON键的动作程序中，通道2用来产生不可屏蔽中断。

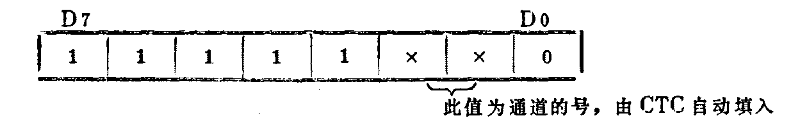
它们都没有使用IM2中断方式，因而也不需要使用中断矢量。

通道3—在本机中用于"LOAD"键的动作程序，该程序将盒式录音机磁带中的信息传送到RAM中。

各个通道的中断服务程序的起始地址表放在ROM中，如下表所示：



CTC四个通道的中断矢量按偶数序排列，中断矢量低字节写入CTC通道0，其格式如下：



通道0的输入和输出都接向布线区以供用户使用。

四、键盘和显示

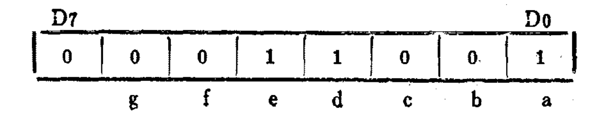
本机有六个七段LED显示器，可以显示十六进制数，也可以显示其它的一些特定字符，在CJBUG监控程序中可显示"—"和"/"和"空"。

所显示的数据(或字符)写入U12的八锁存器74LS273(口地址为88_H)，至于要显示哪一位则由U11单元的八锁存器74LS273(口地址为$8C_H$)来选择。

七段LED显示器的图形以及段号与数据位的对应关系如图3.6所示。LED显示器最左边的一位命令为DG_0位，其余的依次为DG_1，DG_2，DG_3，DG_4，DG_5。例如，要在最左边的LED显示器DG_0位上显示"4"，可以通过下列程序段来完成：

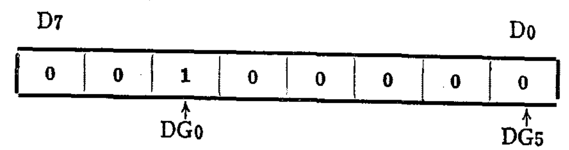
```
LD      A, 19H
OUT     (88), A
LD      A, 20H
OUT     (8C), A
```

其中19_H的二进制表示为：



D7 不起作用，对于 D6—D0 来说 0 为有效，故 b，c，f，g 段亮，得“4”。

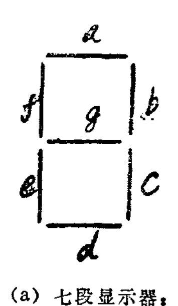
数 20_H 的二进制表示为：



D7，D6不起作用，对于 D5—D0 来说，1 为有效，故显示最左边的 DG0 位。

实际的显示过程并非如此简单。请参考 CJBUG 监控程序中的 DISUP 程序段以及起始地址为 $07A6_H$ 的七段显示表 SEGPT。

晶体管 Q1-Q7用来提高驱动器能力，U1—U3 驱动器用来增加吸收电流的能力。U11 不仅扫描显示，还能扫描键盘。例如，如图 I.1所示，当口(8C)U11中数为08H(即L3线为低电平，其余线为高电平)时可检查 4，5，6，B，MON 键有否被压下。如果键 6 被压下，则 R2 线呈低电平，其余线呈高电平，即口(90)中低五位为11011B。由08H和1011B两个数字可以求出键的偏移量(功能键的键偏移量都大于等于 10_H)。



(a) 七段显示器；

数字	D7 D6 D5 D4 D3 D2 D1 D0 g f e d c b a	字形
0	1000000	0
1	1111001	1
2	0100100	2
3	0110000	3
4	0011001	4
5	0010010	5
6	0000010	6
7	1111000	7
8	0000000	8
9	0011000	9
A	0001000	A
B	0000011	b
C	1000110	C
D	0100001	d
E	0000110	E
F	0001110	F
I	1111101	I
空	1111111	
—	0111111	-

(b) 七段显示的代码与字形的关系

图 3.6

例如 EXEC 键为 10_H，SS 键为 11_H 等，详见起始地址为 07B9H 的键值表 KYTBL。

详细的键盘分析程序 DECKY 请参考附录三的 CJBUG。

五、录音机接口电路(一)—转储(Dump)

将存放在 RAM 中易失的程序或信息转储到廉价的盒式磁带中，通常可以使用市场上出售的廉价的录音机和录音磁带。在转储过程中我们采用了“堪萨斯(Kansas)城标准”的

记录技术。

用于盒式磁带上记录数据的格式必须遵守两个标准：记录“1”和“0”的堪萨斯城标准和记录数据块的 Intel Hex Format。

堪萨斯城标准是 1975 年 9 月 7 日和 8 日在美国堪萨斯城由 BYTE 杂志主持的专题讨论会上制定的。这次专题讨论会的目的是，为业余爱好者的录音磁带记录技术制订标准。堪萨斯城标准的要求是(本机是符合这些规定的)：

1）八个频率为 2400H_z 的周波表示逻辑 1；

2）四个频率为 1200H_z 的周波表示逻辑 0；

3）一个记录字符的构成为：一个逻辑 0 的起始位、7 或 8 个数据位、两个或两个以上的停止位(本机使用一个七位的 ASCII 数据字符和一个停止位)；

4）七个 ASCII 数据位的次序为：最低位在先，最高位在后；

5）在数据块之前有 30 秒以上的全 1 导引段，之后有 5 秒的全1结尾段；

6）数据的传送速率为 300 波特(每位用 3.33ms)；

7）数据块内容不作规定。

因为堪萨斯城标准没有对所记录数据块的内容作出规定，所以选用另一个标准—Intel Hex Format 来规定数据块的结构。该标准的要点是：

1）在数据块内的每个记录以冒号(：)开始，以回车和换行符号结束；

2）一切信息用 ASCII(7 位，无奇偶校验)表示；

3）数据记录格式：

字节 1	冒号(：)分解符
字节 2—3	该记录中二进制的字节数，最多为 16 个二进制字节(即 32 个 ASCII 字节)。
字节 4—5	该记录起始地址的高位字节。
字节 6—7	该记录起始地址的低位字节。
字节 8—9	记录类型：ASCII“00”。
字节 10—11	除了分解符与回车和换行符以外的所有字节的“检查和”，此“检查和”为各二进制字节和的负数。

回车和执行

4）“文件结束”记录格式：

字节 1	冒号(：)分解符
字节 2—3	ASCII“0”
字节 4—5	ASCII“0”
字节 6—7	ASCII“0”
字节 8—9	“01”记录类型：ASCII“01”
字节 10—11	“检查和“

用于转储的接口电路如图 I.1 下部所示。利用程序(详细内容请参考 CJBUG 中起始地址为 0402_H 的 DUMP 键盘动作程序)设定 CTC 通道 1 为定时器工作方式，使 ZC1 产生 4800Hz 或 2400Hz 的脉冲，通过 U14 的分频，即可得到 2400Hz 或 1200Hz 的脉冲，再通过阻容(R33 和 C7)滤波，滤去其中的高频分量，即可接至录音机的 Auxiliary 或 MIC

输入端。

六、录音机接口电路(二)—输入(Load)

从 J1 端输入录音机的脉冲信息，U4 为限幅与整形电路，使 U5 能有一个不畸变的方波输入。J1端输入的脉冲的峰—峰值应为 2V 左右，输入电平的高低可以用 U4 驱动 LED 显示器来指示。U14 和 U15 是频率检测器，用来鉴别 1200Hz 和 2400Hz。当 U7 输出 1200Hz 脉冲(“0”)时，U14 的 $\overline{Q}$ 输出为 0，当 U7 输出 2400Hz 脉冲(“1”)时，U14 的 $\overline{Q}$ 输出为 1。$\overline{Q}$ 的数据由 CPU 通过 U13 来读取，并拼装成一个 ASCII 字符，读入的 ASCII 字符再由程序转换成二进制数。

可见，在 CJBUG 的控制下，Z80—CPU 和 Z80—CTC 能实现 UART (通用异步接收发送器)的功能，以接收异步的串行数据，并形成并行的字，然后存入存储器。

3.6 其他部分

一、单步逻辑

当压下 SINGLE STEP 键后，进入该键的动作程序(起始地址为 $030C_H$)。它将 CTC 通道 2 设定为定时器工作方式，并恢复现场，即从栈(用户寄存器存放区)中弹出各个寄存器的内容，然后执行用户的指令。在开始执行用户指令时，CTC 通道2的 ZC2 输出一个脉冲，这个脉冲通过 U33 和 U34 送向 CPU 的$\overline{NMI}$输入端，因而在这一条用户指令执行完毕后，即返回 CJBUG，执行 0066_H 单元的指令。此后程序段(起始地址为 01CBH) 将关闭 CTC 通道 2，并保护现场。最后显示 PC 和累加器A的内容。

二、中断链路

Z80 系列中 I/O 接口器件内均没有中断控制电路，从而与 Intel 8080/8085 系列不同，不必使用专用的中断控制器件。中断的优先级根据该芯片在链路中的位置来决定。本机中 CTC 具有较高的中断优先级，PIO具有较低的中断优先级。PIO 芯片的 IEO 线接向布线区，供用户再增加 I/O 芯片时使用。

三、复位电路(Reset)

图 3.7 示出了本机所用的复位电路。有两种情况可以产生复位信号：一是上电复位，在刚接通电源时，由于电容 C17 的作用，使 $\overline{RESET}$ 短暂的保持低电平信号零；一是使用 RESET 按钮，在压下此按钮期间，$\overline{RESET}$保持低电平。

四、电压保护电路(任选)

本机使用 7805 集成电路。它的输入应大于 +7V。它的输出将稳定在 +5V。

五、S—100 总线

本机有 S—100 总线插脚。这种接口通常能与静态存储器和 I/O 扩充板兼容。如果接口需要 8080 所特有的控制信号，例如 SYNC，INTA，DBIN，POC，PWR，PRD

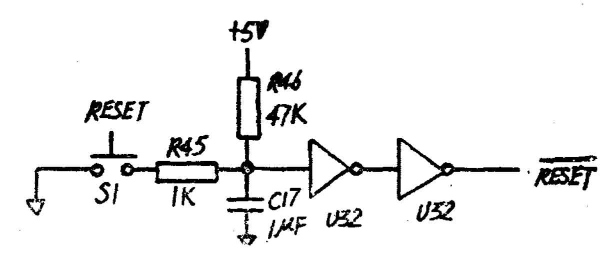


图 3.7 复位电路

等，则尚需在布线区附加一些逻辑电路，以进行信号变换；并/或将附加的信号接到 S—100 总线。在印刷线路板上方有+8V，±18V 电源的接线端，以便接入相应的电源。

六、布线区

布线区供用户附加 25—30 个集成电路，以便扩充存储器、增加 CRT 接口及其他电子实验电路。Z80—CPU，PIO，CTC(部分)的信号线以及译码出来的系统信号都引到布线区的附近，以便用户增加集成电路时使用这些信号。电源线和地线在印刷线路板的背面平行走线，使集成电路紧挨着这一低阻抗的电源，从而减少用户电路的噪声干扰。

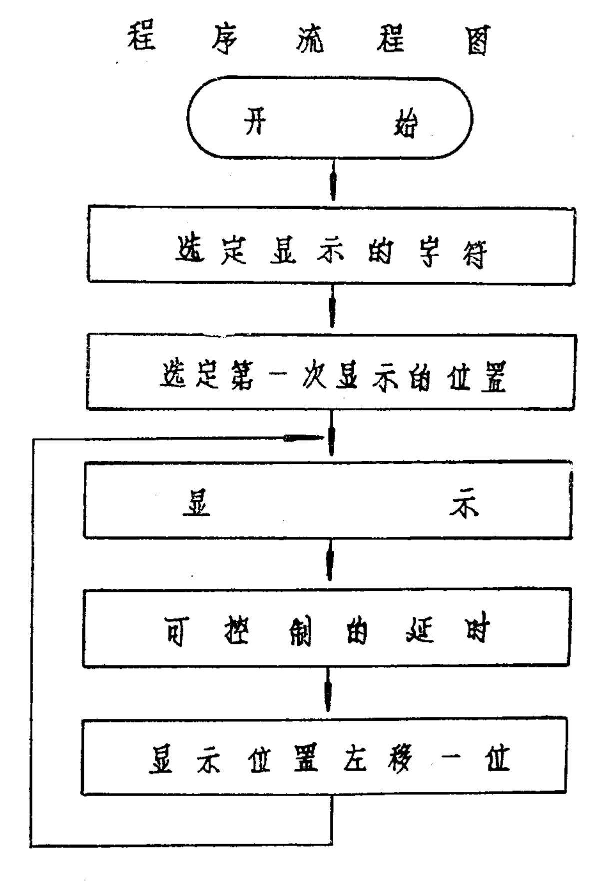
第四章　程序举例

本章目的是想通过几个程序的导入、执行和调试，使用户进一步熟悉键盘的操作功能和本机的使用方法，以及程序设计的一些基本知识。

4.1　软件延时

下面这个程序是使字符“8”，从显示器的右端向左端不停地循环移动。在每个位置上停留的时间，可以由改变装入B寄存器的时间常数来决定，约由20ms到5.2sec。

程 序 流 程 图



；程序

```
                      ORG 2000
2000    3E00          LDA, 00
2002    D388          OUT(88H), A       ；选定字符“8”
2004    3E01          LDA, 01H          ；选定右边第一个显示器亮
2006    D38C       LOOP: (8CH), A
2008    06FF          LDB, FFH          ；B←延时时间常数
```

```
200A    CD6106LOOP1: CALL    D20MS; 调用 CJBUG 中延时约 20ms 子程序
200D    10FB          DJNZ, LOOP1—$; 最长延时约 5.2sec
200F    07            RLCA          ; 左移一位
2010    18F4          JR   LOOP-$   ; 循环回去
```

首先输入程序，按下 MON，输入程序首地址 2000，按下 EXEC，字符"8"则在显示器上从右到左循环移动。按下 MON，程序则停止执行。控制延时子程序 D20MS 的调用次数(也就是控制装入 B 寄存器的时间常数)就可以控制字符在显示器上的停留时间。

4.2 Z80-CTC 的应用

```
                  ORG    2000
2000    3E21      LD     A, 21H
2002    ED47      LD     I, A      ; 设定中断矢量高字节 I→21H
2004    310023    LD     SP,2300H; 建立栈指示器
2007    3E00      LD     A, 00
2009    D384      OUT    (84H), A; 外部设备提供中断矢量低字节
200B    3EA5      LD     A, 0A5H
200D    D384      OUT    (84H), A; 设定 CTC 的工作方式，输入通道控制字
200F    3EFF      LD     A, FFH
2011    D384      OUT    (84H), A; 输入时间常数，延时约 33ms
2013    3E01      LD A, 01H        ; 设定第一次显示 01
2015    ED5E      IM2              ; 设定中断方式 2
2017    FB LOOP:EI
2018    76        HALT
2019    C31720    JP LOOP
        ; 中断服务程序入口地址表
2100    00  22
        ; 中断服务程序
2200    FB        EI               ; 开中断
2201    07        RLC  A           ; A 累加器左移一位
2202    ED4D      RETI             ; 中断返回
```

本例程序是使 Z80-CTC 的 0 通道工作在定时器方式，经过一个整定延迟时间后(本例整定延迟时间约 33ms)，发出一个中断请求。改变输入到 CTC 的时间常数，可以改变请求中断的时间。

输入程序，按以下步骤进行调试：

用 BP 键，在 2200H 设置一个断点。先输入首地址2000H，按下 EXEC 键，立刻显示断点地址 2200H 和这时 A 累加器的内容。连续不断按动 EXEC，A 累加器内容不断变化，而且每按一次，改变显示一次，按 01，02，04，08，10，20，40，80 的顺序改变。

可利用 RESET 键使 CTC 复位。

下面这个程序，是将本例稍加修改后，再与 4.1 例结合起来的一个简单表演程序。

输入程序，按下 RESET 键，输入程序首地址 2000H，按下 EXEC 键，字符"8"快速从显示器右端移动到左端，经过十次循环之后，立刻变为较慢的一次移动，接着又是十次较快的循环移动，又接着一次慢速移动……，这样不停地在显示器上循环移动。根据需要，可以很方便地改变显示的字符和循环的速度。按下 RESET 键，移动立刻停止，回到 CJBUG 控制。

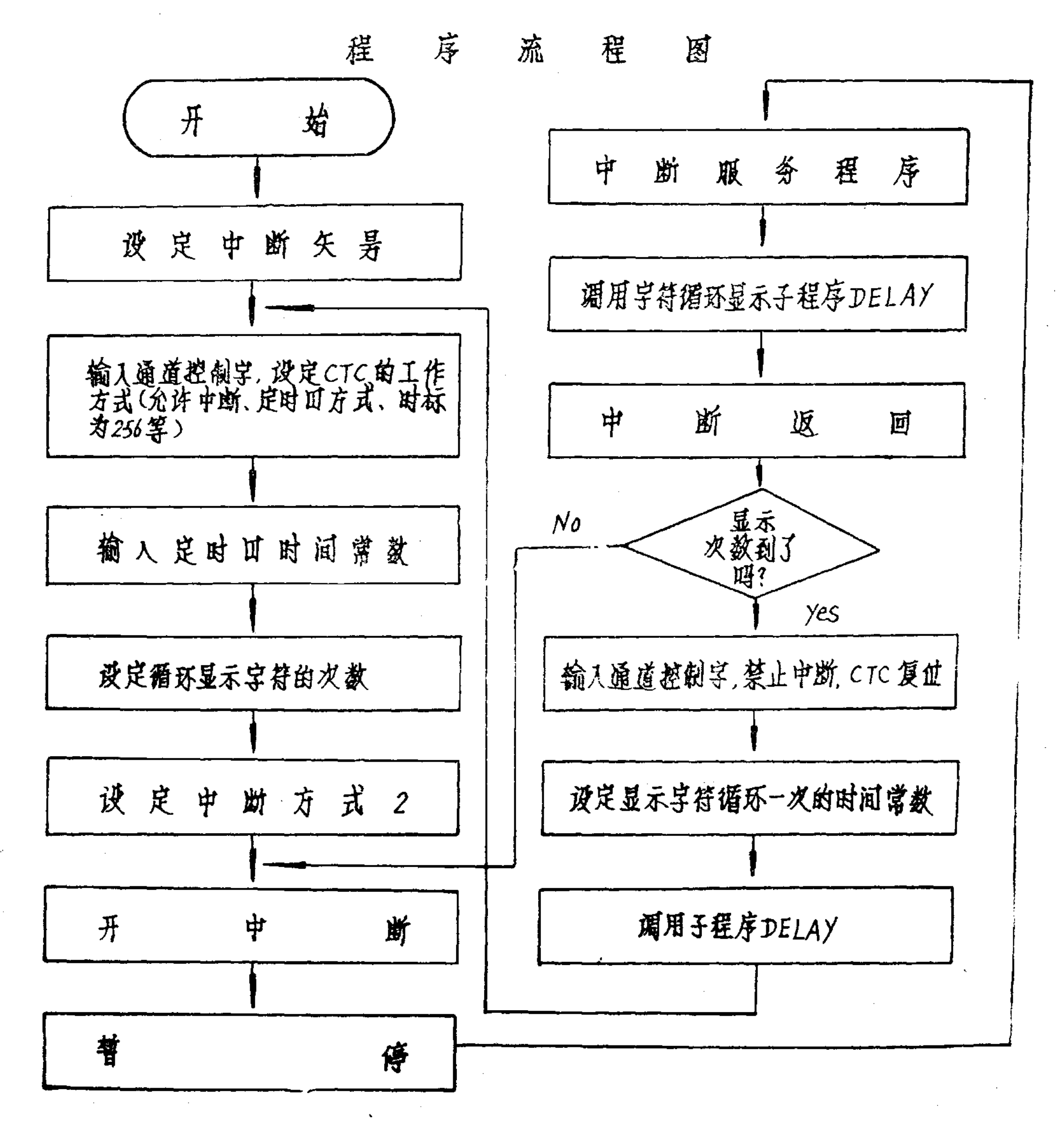
```
      ；程序
                        ORG   2000
2000    3E21          LD      A, 21H
2002    ED47          LD      I, A          ；设定中断矢量高字节
2004    3E00          LD A, 00H
2006    D384          OUT  (84H), A         ；外部设备提供中断矢量低字节
2008    3EA5  ST:     LD     A, 0A5H
200A    D384          OUT  (84H), A         ；输入通道控制字，设定 CTC 工作方式
200C    3EFF          LD A, FFH
200E    D384          OUT (84H), A          ；输入时间常数
2010    0EF6          LD   C, F6H           ；设定快速循环次数(10D 次)
2012    ED5E          IM2                   ；设定中断方式 2
2014    FB LOOP:      EI                    ；开中断
2015    76            HALT
2016    0C            INC  C
2017    20FB          JR  NZ, LOOP-$        ；Z=O 循环返回
2019    3E03          LD A, 03H
201B    D384          OUT  (84H), A         ；禁止通道中断
201D    1E5F          LD   E, 5FH           ；设定字符循环一次的时间常数
201F    CD5020        CALL  DELAY           ；调用子程序
2022    C30820        JP ST                 ；循环返回，重新工作
      ；中断服务程序入口地址表
2100    0022
      ；中断服务程序
2200    1E0A          LDE, 0AH              ；设定字符循环速度的时间常数
2202    CD5020        CALL  DELAY
2205    ED4D          RETI
      ；子程序
2050    3E00DELAY: LDA, 00H; 选定字符"8"
2052    D388          OUT  (88H), A
2054    3E01          LD A, 01H
2056    D38CLOOP1: OUT  (8CH), A
```

```
2058    43          LD   B, E
2059    CD6106LOOP2: CALL  D20MS
205C    10FB        DJNZ, LOOP2-$
205E    CB6F        BIT  5, A        ；测试A的第5位
2060    2003        JR   NZ, LOOP3-$
2062    07          RLCA
2063    18F1        JR   LOOP1-$
2065    C9LOOP3: RET
```

注：(1) 这个[DELAY]子程序，是将4.1例稍加修改而成。这种设计方法，不仅易于调试和检查，而且也节省存储单元。

(2) 若想显示别的字符，可参考下表，修改 DELAY 子程序的第一条指令 LD A，00_H 中的立即数(即修改2051存储单元的内容)。

程 序 流 程 图



显示的字符	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
立即数	40	79	24	30	19	12	02	78	00	18	08	03	46	21	06	0E

4.3 Z80-PIO的应用

首先用一个 10KΩ 电阻，将 Z80—PIO 的 $\overline{ASTB}$脚连接到电源+5V 端，然后输入下列程序，并用 BP 键在 2200_H 单元设置断点。

输入程序首地址 2000_H，按下 EXEC 键，CPU 执行程序到暂停指令 HALT，显示器无显示。

取一导线，使 $\overline{ASTB}$ 脚碰一下地(GND)，相当于输入一负脉冲。此脉冲的上升沿使 PIO 发出一中断请求，$\overline{(INT)}$，CPU 将执行中断服务程序，把由 BP 键设置的断点 2200_H 显示出来。

```
        ；程序
                                  ORG       2000
2000    3E21                      LD        A, 21H
2002    ED47                      LD        I, A          ；设定中断矢量高字节
2004    3E00                      LD        A, 00H
2006    D382                      OUT       (82H), A      ；外部设备提供中断矢量
                                                           低字节
2008    3E4F                      LD        A, 4FH
200A    D382                      OUT       (82H), A      ；对 PIO 的A口设置方
                                                          式1
220C    3E87                      LD        A, 87H
200E    D382                      OUT       (82H), A      ；设置允许中断控制字
2010    ED5E                      1M2                     ；中断方式 2
2012    FB              LOOP:     EI
2013    76                        HALT
2014    C31220                    JP LOOP—$
        ；中断服务程序入口地址表
2100    00      22
        ；中断服务程序
2200    FB      EI
2201    ED4D    RETI                                      ；中断返回
```

可以将 4.4 例的中断服务程序及 DELAY 子程序作为本例的中断服务程序。这时，输入程序之后，按下 RESET 键，输入程序首地址 2000_H，按下 EXEC 键，显示器无显示，将 $\overline{ASTB}$ 脚碰一下地，字符“8”立刻从显示器右端移到左端，最后停留在左端不动。再碰

一次，再重复一次上述实象。若将$\overline{ASTB}$脚碰一下地的操作得不好时，字符“8”会循环移动两次才停留在左端不动。这时，可以用一个10KΩ电阻把$\overline{ASTB}$脚与地连接，然后输入程序，用EXEC键执行程序，然后通过一个3KΩ电阻碰一下+5V端，字符立即从显示器右端移到左端并停留在左端不动。再碰一次，再重复上述现象一次。若条件许可，用TTL电平的单脉冲触发就更好。

4.4 求十进制数的算术和

下列程序是将两个5位十进制数(或和数不大于6位的6位数)进行算术加法运算，两数长度是相同的，和数贮存在指定存储单元，并将和数在显示器上显示出来。

；数据

2030——贮存字节数(两位十进制数为1字节，5位十进制数为3字节)；

2031、2032、2033——贮存被加数，按顺序先贮存低字节数；

2041、2042、2043——贮存加数，按顺序先贮存低字节数；

2051、2052、2053——贮存和数，按顺序先贮存低字节数。

；例子　895867+91787=987654

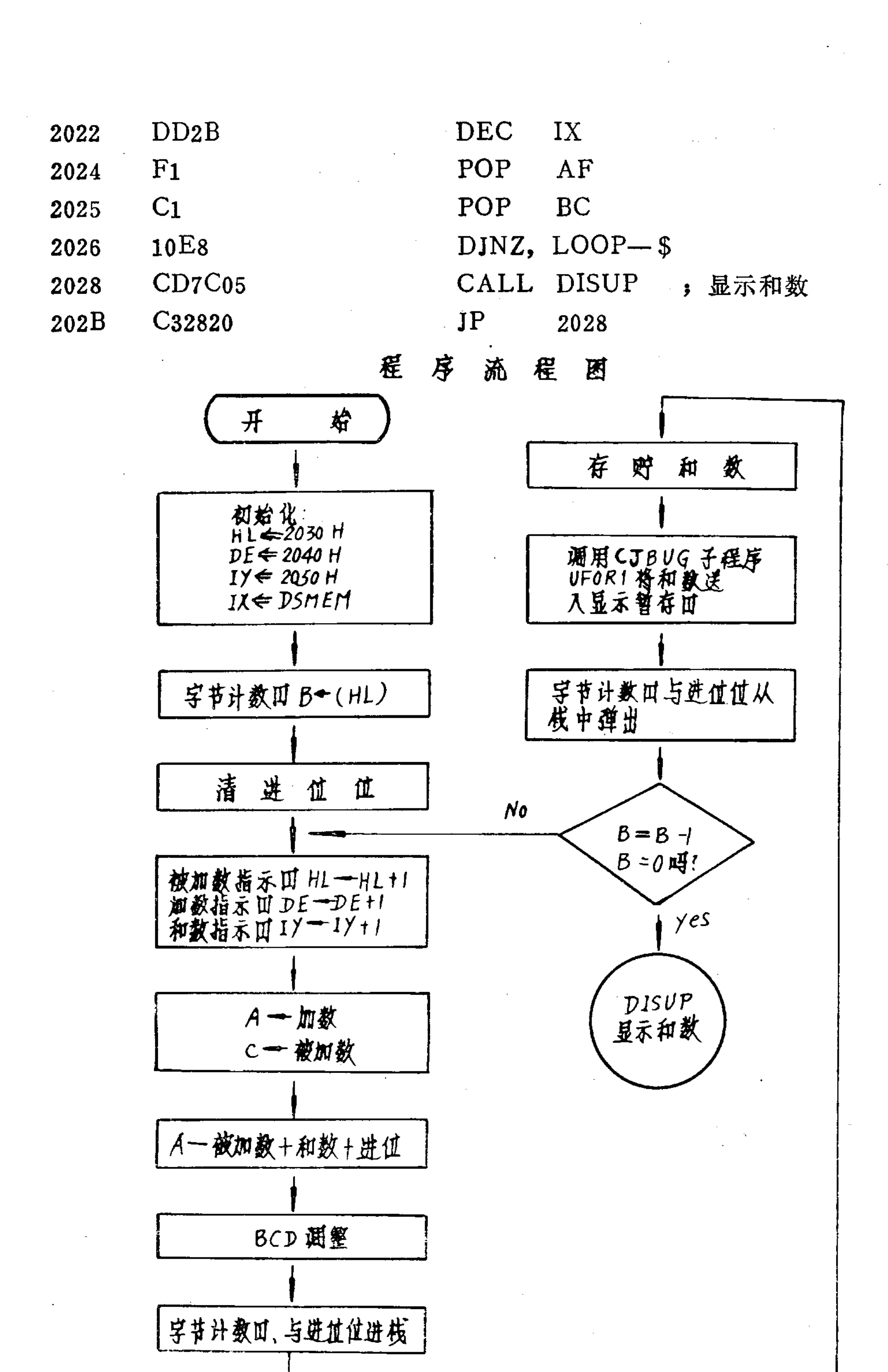
```
字节数：(2030)=03
被加数：(2031)=67    加  数：(2041)=87    和  数：(2051)=
        (2032)=58            (2042)=17            (2052)=
        (0233)=89            (2043)=09            (2053)=
```

；程序

```
2000   213020          LD     HL，2030H  ；初始化
2003   114020          LD     DE，2040H
2006   FD215020        LD     IY，2050H
200A   DD21FB2F        LD     IX，(DSMEM4)
200E   46              LD     B，(HL)    ；计数器←字节数
200F   AF              XOR    A          ；清进位位
2010   23        LOOP：INC    HL         ；指向被加数存储单元地址
2011   13              INC    DE         ；指向加数存储单元地址
2012   FD23            INC    IY         ；指向和数存储单元地址
2014   1A              LD     A，(DE)    ；取出加数
2015   4E              LD     C，(HL)    ；取出被加数
2016   89              ADC    A，C       ；带进位加
2017   27              DAA               ；BCD调整
2018   FD7700          LD     (IY+0)，A；贮存和数
201B   C5              PUSH   BC         ；记忆计数器内字节数
201C   F5              PUSH   AF         ；记忆进位位
201D   CD7906          CALL   UFOR1      ；显示初始化
2020   DD2B            DEC    IX
```

```
2022    DD2B        DEC    IX
2024    F1          POP    AF
2025    C1          POP    BC
2026    10E8        DJNZ,  LOOP—$
2028    CD7C05      CALL   DISUP    ；显示和数
202B    C32820      JP     2028
```

程 序 流 程 图



4.5 时钟计时程序

本程序具有时钟走时功能，显示时、分、秒。

程序设置了三个计时单元：

2102——秒

2103——分

2104——时

执行前可对这三个单元预置任何的十进制数值。当"时"单元(2104)的内容满 60 时，则复为 0。

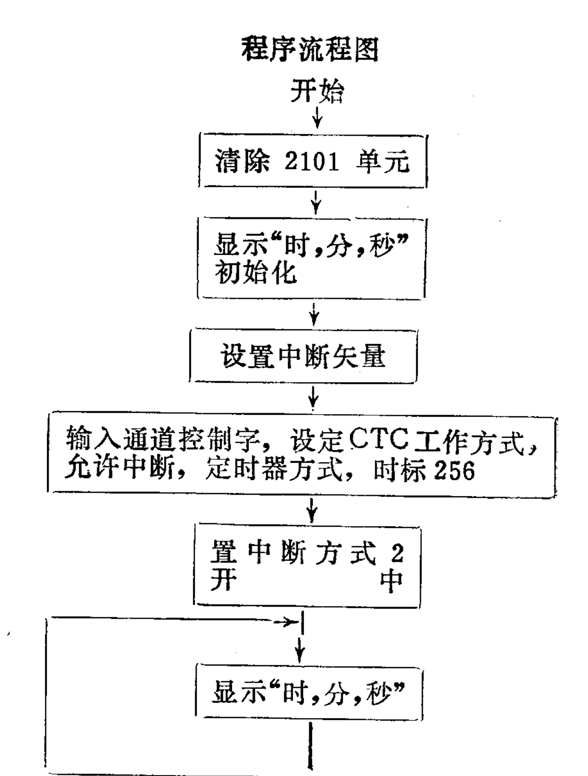
```
        ；程序
2000    AF                  XOR    A
2001    320121              LD     (2101H), A   ；2101 单元清零
2004    ED4B0321            LD     BC, (2103H)
2008    CD7020              CALL   LOOP5        ；显示初始化
200B    3A0221              LD     A, (2102H)
200E    DD21FB2F            LD     IX, 2FFBH
2012    CD7906              CALL   WFOR,        ；显示初始化
2075    3E21                LD     A, 21H
2017    ED47                LD     I, A         ；设置中断矢量高字节
2019    3E10                LD     A, 10H
201B    D384                OUT    (84H), A     ；向外设提供中断矢量低字节
201D    3EA5                LD     A, A5H
201F    D384                OUT    (84H), A     ；输入通道控制字，设(TC)作方式
2021    3E82                LD     A, 82H
2023    D384                OUT    (84H), A     ；输出时间常数
2025    ED5E                IM     2
2027    FB                  EI
2028    CD7C05  LOOP        CALL   DISUP        ；显示
202B    C32820              JP     LOOP
        ；中断服务程序
2030    F5                  PUSH   AF
2031    C5                  PUSH   BC
2032    E5                  PUSH   HL           ；保护寄存器对 BC, AF, HL,
2033    210021              LD     HL, 2100H
2036    0604                LD     B, 04H
2038    37                  SCF
2039    23      LOOP        INC    HL
203A    7E                  LD     A, (HL)
203B    CE00                ADC    A, 00H
203D    27                  DAA
203E    77                  LD     (HL), A
203F    D660                SUB    60H
```

```
2041    3F                    CCF
2042    3001                  JR      NC, LOOP2
2044    77                    LD      (HL), A
2045    10F2      LOOP2       DJNZ    LOOP1
2047    ED4B0321              LD      BC, (2103H)
204B    CD7020                CALL    LOOP5         ；显示初始化
204E    3A0221                LD      A, 0201H
2051    CD9020                CALL    LOOP6         ；显示初始化
2052    E1                    POP     HL
2053    C1                    POP     BC
2054    F1                    POP     AF            ；恢复寄存器对
2055    FB                    EI
2056    ED4D                  RETI
     ；子程序
2070    DD21F72F  LOOP5       LD      IX, DISMEM
2074    78                    LD      A, B
2075    CD7906                CALL    UFOR1
2078    DD23                  INC     IX
207A    DD23                  INC     IX
207C    79                    LD      A, C
207D    CD7906                CALL    UFOR1
2080    C9                    RET
2090    DD21FB2F  LOOP6       LD      IX, DSMEM4
2094    47                    LD      B, A
2095    E60F                  AND     0FH
2097    DD7701                LD      (IX+1), A
209A    78                    LD      A, B
209B    0F                    RRC     A
209C    0F                    RRC     A
209D    0F                    RRC     A
209E    0F                    RRC     A
209F    E60F                  AND     0FH
20A1    DD7700                LD      (IX+0), A
20A4    C9                    RET
```

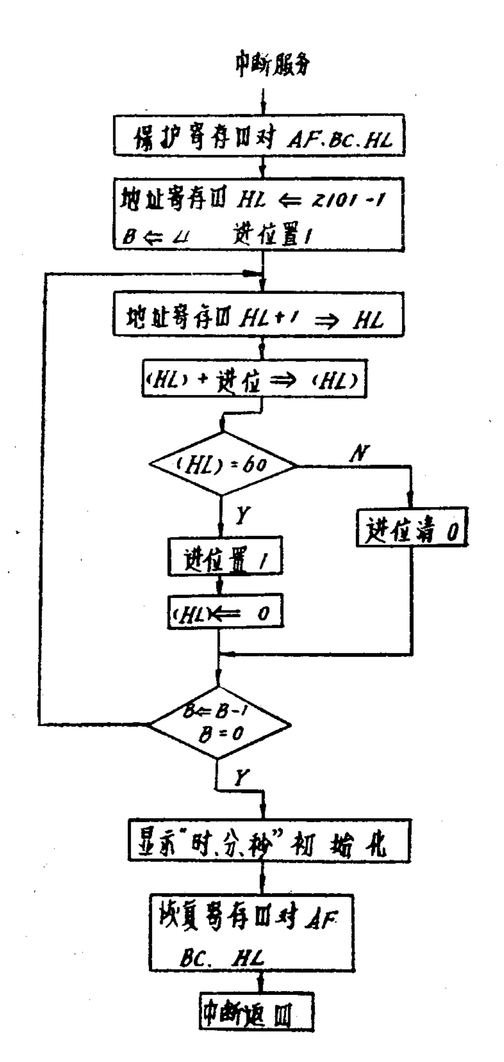
中断服务程序地址

```
2110    30
2111    20
```

程序流程图



开始
清除 2101 单元
显示"时,分,秒"初始化
设置中断矢量
输入通道控制字，设定CTC工作方式，允许中断，定时器方式，时标 256
置中断方式 2 开中
显示"时,分,秒"



中断服务
保护寄存器对 AF.BC.HL
地址寄存器 HL ⇐ 2101-1
B ⇐ 4 进位置 1
地址寄存器 HL+1 ⇒ HL
(HL)+进位 ⇒ (HL)
(HL)=60
N
Y
进位清 0
进位置 1
(HL) ⇐ 0
B ⇐ B-1
B=0
Y
显示"时、分、秒"初始化
恢复寄存器对 AF
BC、HL
中断返回

附　录



目　录

附　录　一

CJ801 原理图、安装位置图、零件表

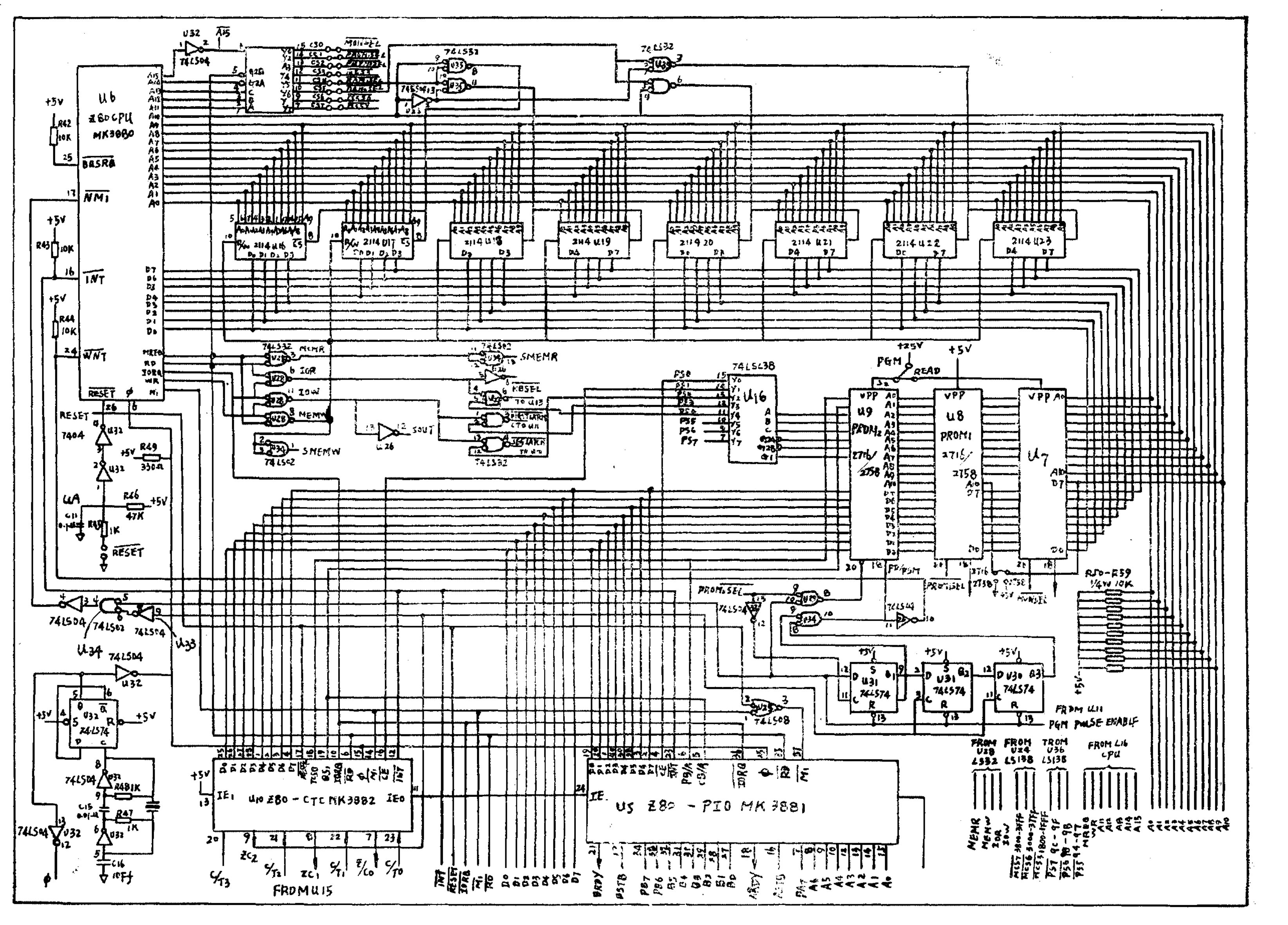


图 1. 1CJ801-Z80 单板计算机线路原理图 1

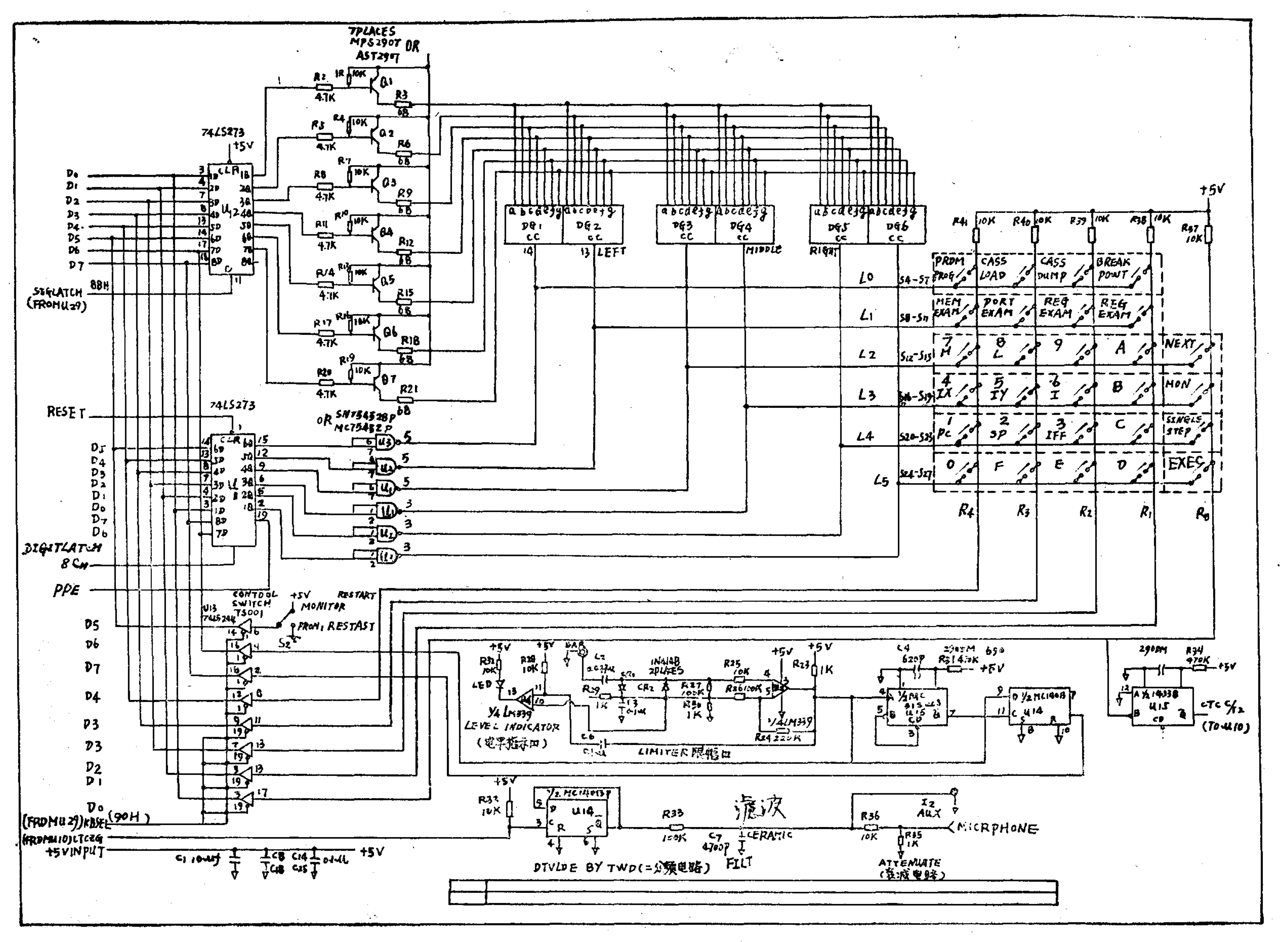


图 1.2 CJ801-Z80 单板计算机线路原理图 2

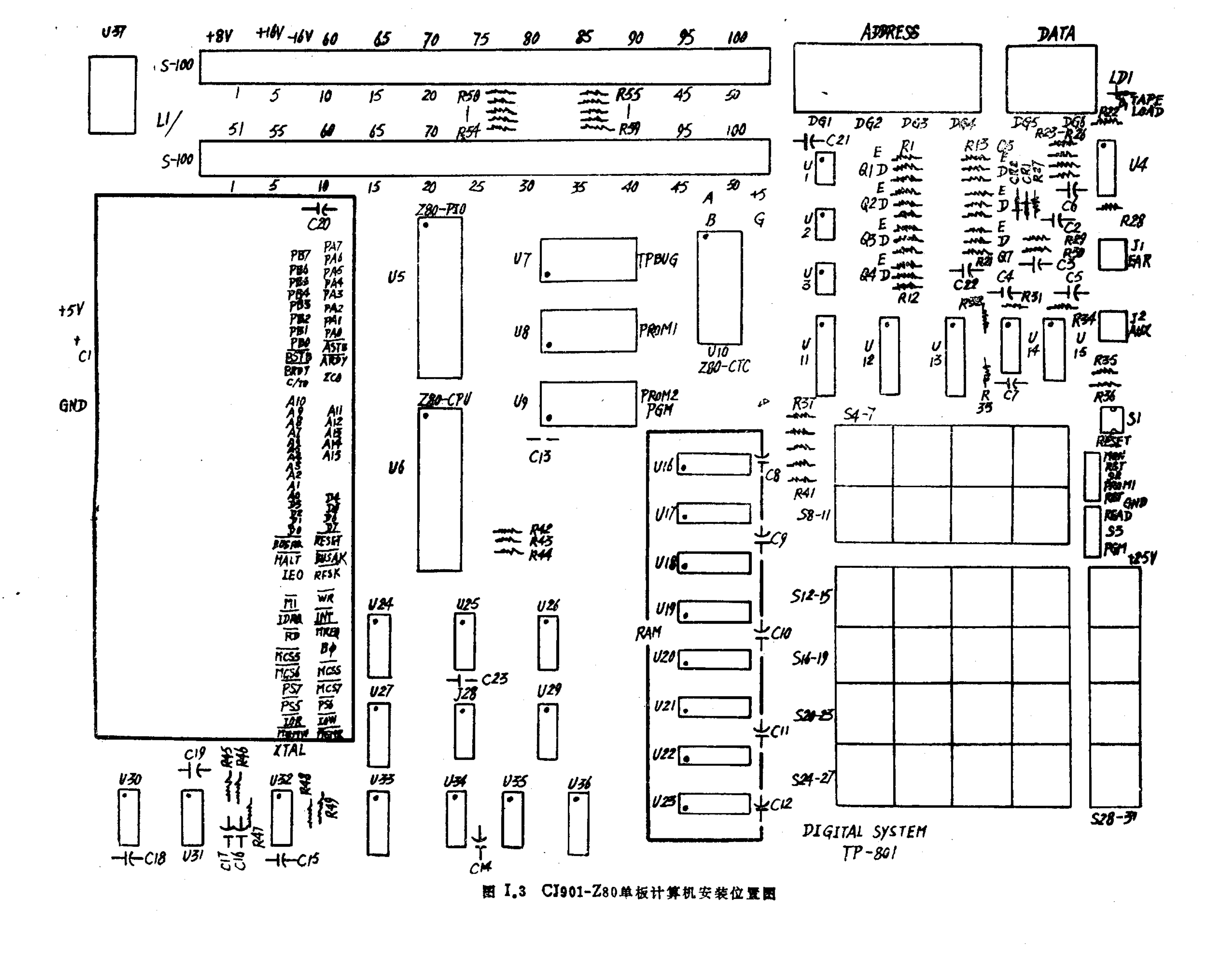


图 I.3 CI901-Z80单板计算机安装位置图

CJ801—Z80 单板计算机零件表

序号	型号与名称（英文）		中文名称	数量	元器件代号
1	Z—80 CPU		中央处理器	1	U6
2	Z—80 PIO		并行输入输出电路	1	U5
3	Z—80 CTC		计数器定时器电路	1	U10
4	2114 RAM		450ns 读写存储器	4/8	U16 U19/U20 U23
5	ROM—TPBUG		只读存储器 TPBUG	1	U7
6	74LS138	Decoder	译码器	2	U24、U36
7	74LS32	Quad OR Gate	四或门	3	U28、U29、U35
8	74LS04	Hex Inverting Buffer	六反相器	2	U26、U33
9	7404	Hex Inverting Buffer	六反相器	1	U32
10	74LS08	Quad And Gate	四与门	1	U25
11	74LS244	Octal Buffer	8位缓存器	1	U23
12	74LS273	Octal Latch	8位D锁存器	2	U11、U12
13	74LS74	Dual Latch	双D锁存器	2	U30、U31
14	74LS02	Quad NOB Gate	四或非门	1	U34
15	75452PP	eripheral Driver	驱动器	3	U1、U2、U3
16	MC14538BCP	CMOS Monostable	CMOS 双单隐态触发器	1	U15
17	MC14013BCP	CMOS Latch	CMOS 双 D 锁存器	1	U14
18	LM339	Quad Comparitor	四比较器	1	U4
19	Sigle Digit Display		数码显示器	6	DG1 DG6
20	9012	PNP Transistor	PNP 三极管	7	Q1 Q7
21	IN4148	Diode	二极管	2	CR1～CR2
22	3.9936MHZ	Parallel Resonant Crgsatl	并联▲振晶体	1	XTAL
23	Red LED	Indicater	红发光二极管	1	LD1
24	4.7K Ohm Resistor 1/4W5%		47KΩ 电阻色标：黄、紫、红	7	R2、R5、R8、R11、R14、R17、R20、R32、R36
25	10K Ohm Resistor 1/4W5%		10KΩ 电阻色标：棕、黑、橙	29	R1、R4、R7、R10、R13、R16、R19、R25、R28、R37、R44、R50、R59
26	68 Ohm Resistor 1/4W5%		68Ω 电阻色标：蓝、灰、黑	7	R3、R6、R9、R12、R15、R18、R21
27	1K Ohm Resistor 1/4W5%		1K 电阻色标：棕、黑、红	7	R23、R29、R30、R35、R45、R47、R48
28	470K Ohm Resistor 1/4W5%		470 电阻色标：黄、紫、黄	2	R31、R34

续表

序号	型号与名称（英文）	中文名称	数量	元器件代号
29	220K Ohm Resistor 1/4W5%	220K 电阻 色标：红、红、黄	1	R24
30	100K Ohm Resistor 1/4W5%	100K 电阻 色标：棕、黑、黄	3	R26、R27、R33
31	300 Ohm Resistor 1/4W5%	330Ω 电阻 色标：橙、橙、黑	1	R49
32	47K Ohm Resistor 1/4W5%	47K 电阻 色标：黄、紫、橙	1	R46
33	100 Ohm Resistor 1/4W5%	100Ω 电阻 色标：棕、黑、棕	1	R22
34	0.1μF Capacitor 20%	0.1μF 电容	15	C3、C6、C8、C14、C18、C23
35	1μF Capacitor 20%	1μF 钽电容	1	C17
36	10μF Capacitor 20%	10μF 钽电容	1	C1
37	0.0047μF Capacitor 20%	0.047μF 电容	1	C7
38	10PF Capacitor 20%	10PF 电容	1	C16
39	0.01μF Capacitor 20%	0.01μF 电容	1	C15
40	620PF Capacitor 5%	620PF 电容	2	C4、C5
41	0.047μF Capacitor 20%	0.047μF 电容	1	C2
42	Push Button Control Switch	按钮	1	S1
43	Slide Switch	控制开关	2	S2、S3
44	Key Switch	键开关	28	S4 S31
45	Key Top with Transparent Cap	带透明罩的键帽	28	
46	Audio jack	录音机插孔	2	J1、J2
47	8Pin IC Socket	8脚插座	3	U1、U2、U3
48	14Pin IC Socket	14脚插座	12	U4、U14、U25、U26、U28、U29 U35
49	16Pin IC Socket	16脚插座	3	U15、U24、U36
50	18Pin IC Socket	18脚插座	4	U16 U19
51	20Pin IC Socket	20脚插座	3	U11、U12、U13
52	24Pin IC Socket	24脚插座	3	U7、U8、U9
53	28Pin IC Socket	28脚插座	1	U10
54	40Pin IC Socket	40脚插座	2	U5、U6
55	Printed Crcuit Board 354×264mm	印刷电路板	1	
56	Plastic Support 15mm	塑料支柱	14	
57	Steel Screw M3×12 S/T R/H	M3×122 螺钉	14	

附　录　二

CJ801 所用集成电路引脚图

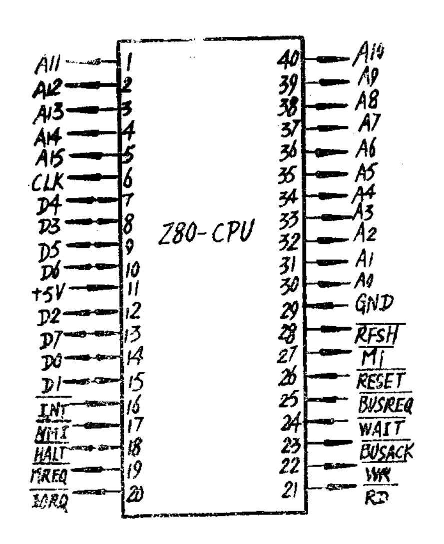
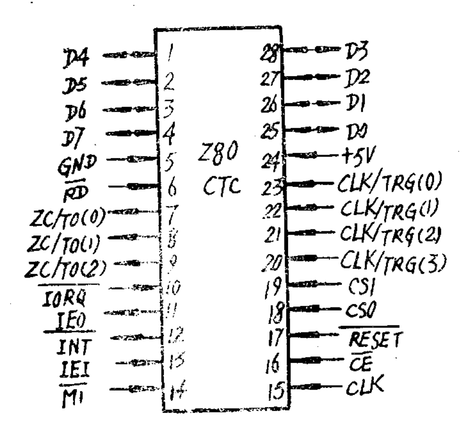


图 II.1 Z—CPU 引脚图



图II. 2 Z80—PIO 引脚图

图 II.3 Z80—CTC 引脚图

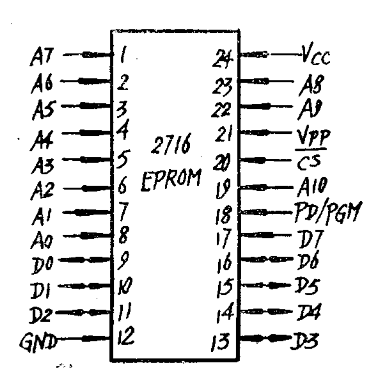


图 II.4 2716 EPROM 引脚图

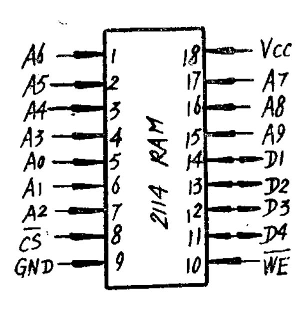


图 II.5 2114 RAM 引脚图

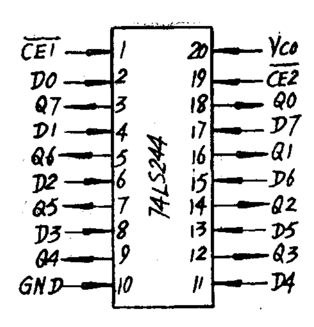


图 II.6 74LS244 八缓存器引脚图

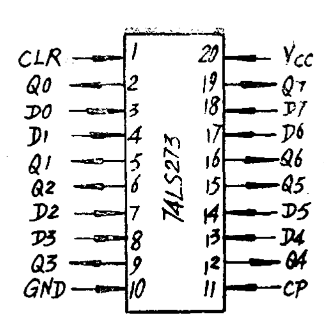
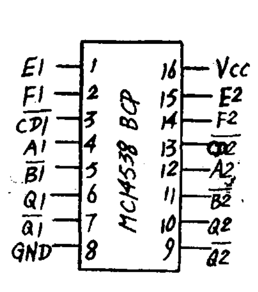


图 II.7 74LS273 八D锁存器引脚图



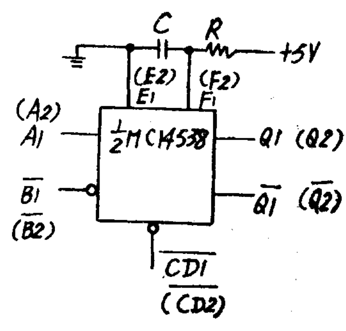
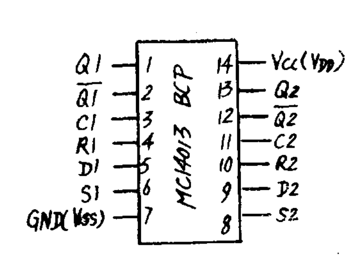


图 II.8 CMOS—MC14538BCP 双单稳态触发器引脚图及框图



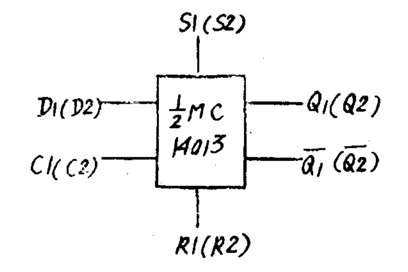


图 II.9 CMOS—MC14013BCP 双 D 锁存器引脚图及框图

G1	G2A	G2B	C	B	A	输出	
1	0	0	0	0	0	$\overline{Y0}=0$	其余为1
1	0	0	0	0	1	$\overline{Y1}=0$	其余为1
1	0	0	0	1	0	$\overline{Y2}=0$	其余为1
1	0	0	0	1	1	$\overline{Y3}=0$	其余为1
1	0	0	1	0	0	$\overline{Y4}=0$	其余为1
1	0	0	1	0	1	$\overline{Y5}=0$	其余为1
1	0	0	1	1	0	$\overline{Y6}=0$	其余为1
1	0	0	1	1	1	$\overline{Y7}=0$	其余为1
不是上述情况			X	X	X	全部输出为1	

图 II.10 74LS138 八中取一译码器引脚图及真值表

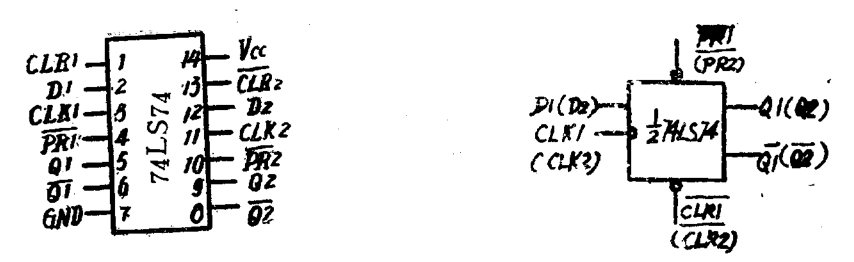


图 II.11　74LS74 双 D 锁存器的引脚图及框图



图 II.12　LM339 四比较器的引脚图



图 II.13　74LS04 六反相器的引脚图



图 II.14　74LS08 四与门的引脚图



图 II.15　74LS32四或门的引脚图

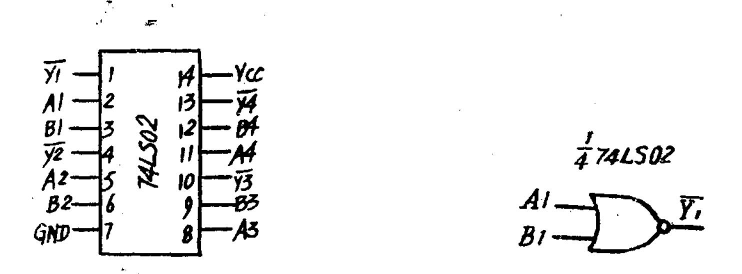


图 II.16　74LS02 四或非门的引脚图



图 II.17　75452P 双驱动器的引脚图

附 录 三

CJBUG 监控程序清单

地址	目的码	标号	指令	注释
0000	AF		XOR A	；A 清零
0001	00		NOP	
0002	31 BA2F		LD SP，$2FBA_H$	；用户栈底为 $2FBA_H$
0005	C3 BB 00		JP RESTR	；转初始化程序
0008	ED73D9 2F	BPENT：	LD(STKPT)，SP	；保存用户的栈指针 SP
000C	F5		PUSH AF	；用户寄存器进栈
000D	00		NOP	
000E	1803		JR $BPENT_2$-$	
0010	C3C42F		JP RST_{16}	
0013	CD8100	$BPENT_2$：	CALL $NMIS_3$	；用户寄存器进栈
0016	1803		JR $BPENT_3$-$	
0018	C3C72F		JP RST24	
001B	DD7EFE	BPENT3：	LD A，(IX-2)	；测试 PC 的低字节
001E	1803		JR BPENT4-$	
0020	C3CA2F		JP RST32	
0023	B7	BPENT4：	OR A	；设置状态标志
0024	200D		JR NZ，BPENT6-$	；PC_L不为0，则PC_L=1
0026	1803		JR BPENT5-$	
0028	C3CD2F		JP RST40	
002B	DD35FF	BPENT5：	DEC (IX-1)	；PC_L为0，则先将高字节减 1
002E	1803		JR BPENT6-$	
0030	C3D02F		JP RST48	
0033	DD35FE	BPENT6：	DEC (IX-2)	；PC_L-1 使其指向断点处的 OP 码
0036	1803		JR BPENT7-$	
0038	C3D32F		JP RST56	

地址	目标码	标号	指令	注释
003B	3ABA2F	BPENT7:	LD A, (SIFLG)	
003E	B7		OR A	
003F	280E		JR Z, BPENT8-$	;是否是 CALL 指令？是转 $004F_H$
0041	3ABB2F		LD A, (OPSAVE)	;取要恢复的操作码
0044	DD66FF		LD H, (IX-1)	
0047	DD6EFE		LD L, (IX-2)	;HL 指向 CALL 指令的下地址
004A	77		LD(HL), A	;恢复 OP 码
004B	AF		XOR A	
004C	32BA2F		LD(SIFLG), A	;清 SIFLG 标志
004F	CDA506	BPENT8:	CALL UFOR3	
0052	2862		JR Z, $BPENT_B$-$	;是否有断点，无断点显示 PC
0054	DD7E02	BPENT9:	LD A, (IX+2)	;从断点表取需恢复的 OP 码
0057	FECF		CP CF_H	;与 CF_H 比较
0059	2807		JR Z, $BPENT_A$-$	;是CF_H，转0062_H
005B	DD6E01		LD L, (IX+1)	
005E	DD6600		LD H, (IX+0)	;取断点地址送 HL
0061	77		LD(HL), A	;恢复操作码
0062	C3B100	$BPENT_A$:	JP $BPENT_A'$	
0065	FF			
0066	ED73D92F	NMI:	LD(STKPT), SP	;保护用户的 SP
006A	F5		PUSH AF	
006B	CD8100		CALL NMIS3	;关 CTC，保存寄存器
006E	3AF12F	NMIS1:	LD A, (SSFLG)	;是单步方式吗？
0071	B7		OR A	;设置状态标志
0072	CAEE00		JP Z, DECKY	;否，由 MON 键发出的中断

地址	目的码	标号	指令	注释
0075	AF	NMIS2:	XOR A	
0076	32F12F		LD(SSFLG), A	；清 SSFLG 标志
0079	CDA506		CALL UFOR3	；取断点数目
007C	2838		JR Z, $BPENT_B$-$	；无断点转去显示 PC 和 A
007E	C3E802		JP CCS_1C	；装配断点
0081	F1	NMIS3:	POP AF	；将 CALL 指令返回地址送 AF
0082	C5		PUSH BC	
0083	F5		PUSH AF	；用户寄存器进栈
0084	C1		POP BC	；CALL 返回地址送 BC
0085	3E03		LD A, 03_H	
0087	D386		OUT(CTC2), A	；关 CTC
0089	ED57		LD A, I	；中断允许触发器 IFF 送寄存器 F
008B	F3		DI	；关中
008C	D5		PUSH DE	
008D	E5		PUSH HL	
008E	F5		PUSH AF	；保护 I 和 IFF
008F	08		EX AF, AF′	；取辅助寄存器内容
0090	D9		EXX	
0091	F5		PUSH AF	
0092	C5		PUSH BC	
0093	D5		PUSH DE	
0094	E5		PUSH HL	
0095	DDE5		PUSH IX	
0097	FDE5		PUSH IY	
0099	DD2AD92F		LD IX, (STKPT)	；取用户栈指针 SP

地址	目标码	标号	指令	注释
009D	DD23		INC IX	
009F	DD23		INC IX	
00A1	DD22D92F		LD(STKPT),IX	；将 $2FBA_H$ 送用户堆栈指示器
00A5	DD7EF4		LD A, (IX−12)	；堆栈中的 IFF送 A
00A8	E604		AND 04_H	；保留 F 中的 IFF，屏蔽掉其它位
00AA	32EE2F		LD(UIF), A	；保护 IFF
00AD	D9		EXX	
00AE	C5		PUSH BC	
00AF	D9		EXX	；将 CALL 返回地址再次压入
00B0	C9		RET	；返回
00B1	CD5906	$BPENT'_A$:	CALL UIX3	；取下一个断点地址，断点计数器 B 减 1
00B4	209E		JR NZ, $BPENT_9$−$	；再次返回，直到恢复完
00B6	CD6505	$BPENT_B$:	CALL DIUPCA	
00B9	1835		JR $DECKY_1$−$	；显示 PC 和 A
00BB	ED73D92F	RESTR:	LD(STKPT), SP	；(STKPT) = $2FBA_H$
00BF	31A22F		LD SP, $2FA2_H$	；系统栈底为 $2FA2_H$
00C2	32DD2F		LD (BFLG), A	
00C5	32EE2F		LD(UIF), A	
00C8	32BA2F		LD(SIFLG), A	；清状态标志
00CB	3E11	RESTR1:	LD A, 11_H	；提示符"—"序号送A
00CD	32FF2F	RESTR2:	LD(SMON), A	；A 的内容送换挡标志
00D0	CD6B06	RESTR3:	CALL UFGCR	；清标志， (KEYPTR) = $2FF7_H$
00D3	3AFF2F		LD A, (SMON)	；换挡标志的内容送 A
00D6	21F72F		LD HL, DISMEM	；HL 指向 DISMEM
00D9	77		LD(HL), A	；(HL)←A

地址	目标码	标号	指令	注释
00DA	3E10		LD A, 10_H	；熄灭符序号送 A
00DC	0605		LD B, 05_H	
00DE	23	RESTR4:	INC HL	
00DF	77		LD(HL), A	；熄灭符序号 10_H 送 $DSMEM_{1-5}$
00E0	10FC		DJNZ $RESTR_4$-$	
00E2	DB90		IN A, (KBSEL)	
00E4	CB6F		BIT 5, A	
00E6	2003		JR NZ, DISUP-$	；检测开关 S_2 的位置处在 MON 则转 $00EB_H$，进入显示程序
00E8	C30008		JP 0800_H	；S_2 处在 PROM 转 0800_H
00EB	CD7C05	DISUP:	CALL DISUP0	；显示
00EE	3E7F	DECKY:	LD A, $7F_H$	为了避免在键盘扫描时可能引起的闪烁，全
00F0	D388	CECKY1:	OUT (SEGLH), A	；熄灭符送 88_H。
00F2	3E3F		LD A, $3F_H$	
00F4	D38C		OUT (DIGLH), A	；将键的所有行控制线接为低电平
00F6	DB90		IN A, (KBSEL)	
00F8	E61F		AND $1F_H$	
00FA	FE1F		CP $1F_H$	；输入列数据与 $1F_H$ 比较
00FC	28ED		JR Z, DISUP-$	；无键按下返回显示
00FE	CD6106		CALL D20MS	；有键按下延时 20ms 消去键抖动
0101	0E8C		LD C, $8C_H$	；设备号 $8C_H$ 送 C
0103	0601		LD B, 01_H	；设置扫描显示的最低位
0105	ED41	KEYDN1:	OUT(C), B	；选中其中的一行键
0107	DB90		IN A, (KBSEL)	
0109	E61F		AND $1F_H$	

地址	目的码	标号	指令	注 释
010B	FE1F		CP 1FH	；输入列数据与 1FH 比较
010D	2009		JR NZ, KEYDN2-$	；按下的键找到转 0118H
010F	CB20		SLA B	；否，选择上一行
0111	3E40		LD A, 40H	
0113	B8		CP B	
0114	20EF		JR NZ, KEYDN1-$	；六行键是否已全描完了？否，转 0105H
0116	18D3		JR DISUP-$	；是，转显示(键抖动引起的误判)
0118	4F	KEYDN2:	LD C, A	；列值→C
0119	AF		XOR A	
011A	3D	KEYDN3:	DEC A	
011B	CB38		SRL B	
011D	20FB		JR NZ, KEYDN3-$	；00-行号→A B = 0
011F	CD3E06		CALL ALTER6	；(00-行号)＊10→A
0122	81		ADD A, C	；产生键值 = (00 - 行号)10H + 列值
0123	21B907		LD HL, KYTBL	；设置表指针 HL = 07B9H
0126	BE	KEYDN4:	CP (HL)	
0127	2809		JR Z, KEYDN5-$	；在表中找到等于A的项，转 0132H
0129	23		INC HL	；否，键值表指针加 1
012A	04		INC B	；B 增量以建立键序值
012B	C33001		JP KEYDN4′	
012E	00		NOP	
012F	00		NOP	
0130	18F4	KEYDN4′	JR KEYDN4-$	；转 0126H 继续找
0132	DB90	KEYDN5:	IN A, (KBSEL)	
0134	E61F		AND 1FH	

地址	目的码	标号	指令	注　释
0136	FE1F		CP $1F_H$	
0138	20F8		JR NZ, KEYDN5-$	；检测键释放
013A	CD6106		CALL D20MS	；为消去键抖动而延时 20ms
013D	78		LD A, B	
013E	FE10		CP 10_H	
0140	3043		JR NC, KEYDN6-$	；键序值与 10_H 比较≥10_H 为命令键转0185H
0142	2ADB2F		LD HL, (KEYPTR)	
0145	70		LD (HL), B	；键序值送入显示缓冲器
0146	B7		OR A	；清进位
0147	01F82F		LD BC, DSMEM1	
014A	ED42		SBC HL, BC	
014C	2820		JR Z, KEYDN8-$	；输入的是第 2 位数吗？转 016EH
014E	B7		OR A	；清进位
014F	01FA2F		LD BC, DSMEM3	
0152	2ADB2F		LD HL, (KEYPTR)	
0155	ED42		SBC HL, BC	
0157	281A		JR Z, KEYDN9-$	；输入的是第4位数吗？转0173H
0159	B7		OR A	；清进位
015A	01FE2F		LD BC, DSMEM7	
015D	2ADB2F		LD HL, (KEYPTR)	
0160	ED42		SBC HL, BC	
0162	2815		JR Z, KEYDNA-$	；输入的是第 8 位数吗？即是否满修改标志，是转 0179_H
0164	2ADB2F	KEYDN7:	LD HL, (KEYPTR)	
0167	23		INC HL	

地址	目的码	标号	指令	注　释
0168	22DB2F		LD(KEYPTR), HL	；指针加1回送
016B	C3EB00		JP DISUP	；转显示
016E	21EF2F	KEYDN8:	LD HL, DIG2	
0171	1803		JR KEYDN9′-$	；树2位数标志
0173	21F02F	KEYDN9:	LD HL, DIG4	
0176	34	KEYDN9′	INC(HL)	
0177	18EB		JR KEYDN7-$	；树4位数标志
0179	CDA905	KEYDNA:	CALL ALTER	；修改处理
017C	2ADB2F		LD HL, (KEYPTR)	
017F	2B		DEC HL	；为再修改将指针调到第六位数位置
0180	22DB2F		LD(KEYPTR), HL	
0183	18E6		JR DISUP-$	；转显示
0185	D610	KEYDN6:	SUB 10H	；项号=序号-10H
0187	4F		LD C, A	
0188	81		ADD A, C	
0189	81		ADD A, C	；偏移量=3*(项号)
018A	4F		LD C, A	
018B	0600		LD B, 00H	
018D	3AFF2F		LD A, (SMON)	；取换档标志
0190	CB47		BIT 0, A	
0192	200A		JR NZ, KEYDN6A-$	；(2FFF)=11 下档命令 转KEYDN6A
0194	79		LD A, C	
0195	FE0A		CP 0AH	；再判是否是单键，是转 KEYDN6A
0197	3805		JR C, KEYDN6A-$	
0199	21C001		LD HL, JPTAB′	；上档命令转移表首址送HL

地址	目的码	标号	指令	注释
019C	09		ADD HL, BC	；表地址＝首址＋偏移量
019D	E9		JP (HL)	；PC←表地址
019E	21A301	KEYDN6A：	LD HL, JPTAB	；下档命令转移表首址送HL
01A1	09		ADD HL, BC	；项号地址＝表首址＋偏移量
01A2	E9		JP (HL)	；PC←表地址
01A3	C3C702	JPTAB：	JP CCS1	；EXEC处理
01A6	C30C03		JP CCS2	；SS 处理
01A9	C3CB00		JP RESTR1	；MON 处理
01AC	C3C701		JP CCS4	；MON′处理
01AF	C3E401		JP CCS5	；NEXT处理
01B2	C31A02		JP CCS6	；LAST处理
01B5	C34E02		JP CCS7	；REG 处理
01B8	C3B802		JP CCS8	；MEM 处理
01BB	C36803		JP CCS9	；BP 处理
01BE	C3A302		JP CCS10	；PORT处理
01C1	C31F02		JP CCS11	；INSERT 处理
01C4	C34203	CCS4：	JP CCS12	；SI处理
01C7	3E12		LD A, 12H	；换挡标志 12H 送 A
01C9	C3CD00		JP RESTR2	；转初始化
01CC	C35D03		JP CCS13	；2FBC 处理
01CF	C36303		JP CCS14	；2FBE 处理
01D2	C39102		JP CCS15	；REG′ 处理
01D5	C3A803		JP CCS16	；TROM处理
01D8	C3BA04		JP CCS17	；LOAD处理
01DB	C30204		JP CCS18	；DUMP处理

地址	目的码	标号	指令	注释
01DE	C31F02		JP CCS19	；DLETE 处理
01E1	C3C203		JP CCS20	；PROM处理
01E4	3AF22F	CCS5:	LD A, (PRFLG)	
01E7	B7		OR A	
01E8	C22605		JP NZ, CCS20D	；是 EPROM 写入方式有效?
01EB	CD8C06		CALL UFOR2	；将输入的四位数送 HL
01EE	3AF32F		LD A, (PFLG)	
01F1	B7		OR A	
01F2	2019		JR NZ, CCS5D=$	；是口检查方式转 CCS5D
01F4	23		INC HL	；HL←HL+1
01F5	23		INC HL	；HL←HL+1
01F6	2B	CCS5A:	DEC HL	；HL←HL-1 指针调整
01F7	3AF42F		LD A, (MFLG)	
01FA	B7		OR A	；是存储器检查方式吗?
01FB	CACB00		JP Z, RESTR1	；不是转向初始化
01FE	7C	CCS5B:	LD A, H	
01FF	CD7906		CALL UFOR1	
0202	7D		LD A, L	
0203	CD5106		CALL IXINC2	；更新存储器地址送 HL
0206	7E		LD A, (HL)	
0207	CD5106		CALL IXINC2	；将新的存储单元内容送 DSMEM4.5
020A	C3EB00	CCS5C:	JP DISUP	；转显示
020D	4C	CCS5D:	LD C, H	
020E	0C		INC C	
020F	79		LD A, C	

地址	目的码	标号	指令	注释
0210	CD7906		CALL UFOR1	；将下一个通道的地址送 DSMEM0.1
0213	ED78		1N A，(C)	
0215	CDCC05		CALL ALTER1	；读新的口子内容，送 DSMEM4.5
0218	18F0		JR CCS5C-$	；转显示
021A	CD8C06	CCS6：	CALL UFOR2	；从 DSMEM0～DSMEM3取数送 HL
021D	18D7		JR CCS5A-$	；转01F6H
021F	3AF02F	CCS11.19：	LD A，(DIG4)	
0222	B7		OR A	
0223	CAD000		JP Z，RESTR3	；是否已输入四位数？没有输入转 00DOH
0226	CD8C06		CALL UFOR2	；从显示缓冲区取地址送 HL
0229	E5		PUSH HL	
022A	D1		POP DE	；DE←HL
022B	D5		PUSH DE	
022C	21012F		LD HL，2F01H	；HL←RAM末址
022F	AF		XOR A	；CY 清零
0230	ED52		SBC HL，DE	
0232	E5		PUSH HL	
0233	C1		POP BC	；传送字节送 BC
0234	3AFF2F		LD A，(SMON)	
0237	CB47		BIT 0，A	
0239	2008		JR NZ，CCS11A-$	；是 INSERT 键转 0243H
023B	E1	CCS19A：	POP HL	；是 DELETE 键
023C	E5		PUSH HL	
023D	23		INC HL	；RAM首址＋1→HL
023E	EDB0		LDIR	；实现搬家(自上而下)

地址	目的码	标号	指令	注释
0240	E1		POP HL	；恢复首址
0241	1809		JR CCS11B－$	；转024C_H
0243	ED5A	CCS11A：	ADC HL，DE	
0245	E5		PUSH HL	
0246	D1		POP DE	；RAM 末地址送 DE
0247	2B		DEC HL	；末址－1→HL
0248	EDB8		LDDR	；实现搬家（自下而上）
024A	E1		POP HL	
024B	23		INC HL	；首址＋1→HL
024C	18B0	CCS11B：	JR CCS5B－$	；转01FEH显示 HL 及（HL）
024E	3E01	CCS7：	LD A，01_H	
0250	32F52F		LD(RFLG)，A	；置寄存器检查标志
0253	21D507		LD HL，$07D5_H$	；HL 放寄存器表首址
0256	CD2506		CALL ALTER5	；找寄存器在堆栈中的位置→HL
0259	7B		LD A，E	
025A	FE06		CP 06_H	
025C	3806		JR C，CCS7A－$	；寄存器键号与 6 比较小于 6 转0264_H
025E	7E	CCS70：	LD A，(HL)	
025F	CDCC05		CALL ALTER1	
0262	1824		JR CCS7D－$	；寄存器内容送 DSMEM4.5
0264	FE03	CCS7A：	CP 03_H	
0266	281A		JR Z，CCS7C－$	；是键 3 吗？是为IFF转 0282_H
0268	FE02		CP 02_H	
026A	2003		JR NZ，CCS7B－$	；是键SP吗？不是转 $026F_H$
026C	21D92F		LD HL，STKPT	；HL指向STKPT

地址	目的码	标号	指令	注释
026F	7E	CCS7B:	LD A, (HL)	
0270	CDCC05		CALL ALTER1	;以用户栈指示器直接取SP_L→DSMEM4.5
0273	DD22DB2F		LD(KEYPTR), IX	;设指针，以便修改程序作统一处理
0277	23		INC HL	;HL指向STKPT:
0278	7E		LD A, (HL)	
0279	DD21F92F		LD IX, DSMEM2	
027D	CD7906		CALL UFOR1	;SP_H→DSMEM2.3
0280	180C		JR CCS7E-$	;转028E_H显示 SP
0282	3AEE2F	CCS7C:	LD A, (UIF)	
0285	CDCC05		CALL ALTER1	;从UIF 取值 0 或 4 送 DSMEM4.5
0288	21FD2F	CCS7D:	LD HL, DSMEM6	
028B	22DB2F		LD (KEYPTR), HL	
028E	C3EB00	CCS7E:	JP DISUP	;设置指针，以便更换 8bit 寄存器
0291	3E01	CCS15:	LD A, 01_H	
0293	32F62F		LD(ARFLG), A	;置辅助寄存器检查标志
0296	21E507		LD HL, 07E5	;辅助寄存器表首址送 HL
0299	CD2506		CALL ALTER5	;找R′在堆栈中的位置送HL
029C	3E12		LD A, 12_H	
029E	DD7701		LD(IX+01), A	;“′”序号送DSMEM1
02A1	18BB		JR CCS70-$	;转 025E_H
02A3	3AEF2F	CGS10:	LD A, (DIG2)	
02A6	FE01		CP 01_H	
02A8	20E4		JR NZ, CCS7E-$	;是否已输入 2 位数，没有转028E_H
02AA	32F32F		LD (PFLG), A	;置口检查标志
02AD	CD8C06		CALL UFOR2	;从 DSMEM0～DSMEM3 取数送 HL

地址	目的码	标号	指令	注释
02B0	4C		LD C, H	
02B1	ED78		IN A, (C)	；口内容送 DSMEM4.5
02B3	CDCC05	CCS10A:	CALL ALTER1	
02B6	18D0		JR CCS7D-$	；转 0288_H
02B8	3AF02F	CCS8:	LD A, (DIG4)	
02BB	B7		OR A	
02BC	28D0		JR Z, CCS7E-$	；是否已输入四位数？没有转 028E
02BE	32F42F		LD(MFLG), A	；置存贮器检查标志
02C1	CD8C06		CALL UFOR2	；从 DSMEM0～DSMEM3 取数据送 HL
02C4	7E		LD A, (HL)	；(HL)→A
02C5	18EC		JR CCS10A-$	；转 $02B3_H$
02C7	AF	CCS1:	XOR A	
02C8	D38C		OUT(DIGLH), A	；关显示
02CA	3AF02F		LD A, (DIG4)	
02CD	B7		OR A	
02CE	2806		JR Z, CCS_1A	；设置状态标志，转到继续执行方式
02D0	CD8C06		CALL UFOR2	；将输入的四位数作为起始地址送 HL
02D3	CDDD02	CCS1A0:	CALL CCS_1B	；更换 PC 值，以便 RET 进入用户程序
02D6	CDA506	CCS_1A:	CALL UFOR3	；测试断点是否有效，
02D9	2035		JR NZ CCS2A-$	；有效，先单步一次转 0310_H
02DB	1821		JR CCS_1E-$	；无效，转 $02FE_H$
02DD	DD2AD92F	CCS_1B:	LD IX, (STKPT)	
02E1	DD75FE		LD(IX-2), L	
02E4	DD74FF		LD(IX-1), H	
02E7	C9		RET	；将 HL 送用户栈的 PC

地址	目的码	标号	指令	注释
02E8	DD21DE2F	CCS_1C:	LD IX, BPTAB	;断点表指针指向 $2FDE_H$
02EC	DD6600	CCS_1D:	LD H, (IX+0)	
02EF	DD6E01		LD L, (IX+1)	;取操作码的地址
02F2	7E		LD A, (HL)	;从用户程序断点处取操作码
02F3	0ECF		LD C, CF_H	
02F5	71		LD(HL), C	;将 CF 送入用户程序断点处
02F6	DD7702		LD(IX+2), A	;保存用户操作码
02F9	CD5906		CALL UIX3	;指向断点表下一个单元
02FC	20EE		JR NZ CCS_1D-$	;为处理更多的断点而循环返回
02FE	3E04	CCS1E:	LD A, 04_H	
0300	D38C		OUT(DIGLH), A	;把 MON 键所在行接为低电平，允许 $\overline{NMI}$
0302	3E45		LD A, 45_H	
0304	D386		OUT (CTC2), A	
0306	3E01		LD A, 01_H	;CTC 计数方式工作，计数次数为 1 次，接受来自 MON 发出的 ZC0 从而产生 $\overline{INT}$
0308	D386		OUT(CTC2), A	
030A	1811		JR CCS2B-$	;恢复用户寄存器
030C	AF	CCS2:	XOR A	
030D	32DD2F		LD(BFLG), A	;清断点数标志
0310	3E01	CCS2A:	LD A, 01_H	
0312	32F12F		LD(SSFLG), A	;设置单步方式标志
0315	3E07		LD A, 07_H	
0317	D386		OUT(CTC2), A	
0319	3E0B		LD A, $0B_H$	;CTC2 以定时方式工作，隔 176T 发 ZC2 引起 NMI，禁止发 $\overline{INT}$
031B	D386		OUT(CTC2), A	
031D	FDE1	CCS2B:	POP IY	

地址	目的码	标号	指令	注释
031F	DDE1		POP IX	(注：在初始化程序中，已将SP指向2FA2即系统栈底)
0321	E1		POP HL	
0322	D1		POP DE	
0323	C1		POP BC	
0324	F1		POP AF	
0325	08		EX AF, AF′	
0326	D9		EXX	
0327	F1		POP AF	
0328	ED47		LD I, A	
032A	E1		POP HL	
032B	D1		POP DE	
032C	C1		POP BC	；恢复寄存器
032D	3AEE2F		LD A, (UIF)	；取 IFF
0330	B7		OR A	；设置状态
0331	C23703		JP NZ, CCS2C	；IFF = 1转0337_H
0334	F1		POP AF	；此时SP已指向$2FB8_H$
0335	F3		DI	；关中
0336	C9		RET	；返回(进入用户程序)
0337	F1	CCS2C：	POP AF	
0338	FB		EI	
0339	C9		RET	；开中返回(进入用户程序)
033A	CDAF06	CCS2D：	CALL UFOR4	
033D	3E01		LD A, 01_H	
033E	C9		RET	
033F	00		NOP	

地址	目的码	标号	指令	注释
0340	00		NOP	
0341	00		NOP	
0342	CD8C06	CCS12:	CALL UFOR2	；从 DSMEM0～DSMEM3→HL
0345	CDDD02		CALL CCS1B	；将 HL 送用户栈 PC 位置
0348	7E		LD A，(HL)	
0349	FECD		CP CD_H	
034B	20C3		JR NZ，CCS2A-$	；当前指令是 CALL？不是转单步处理
034D	23		INC HL	
034E	23		INC HL	
034F	23		INC HL	；HL 指向下一条指令地址
0350	7E		LD A，(HL)	
0351	32BB2F		LD (OPSAVE)，A	；保存下一条指令的第一个操作码
0354	36CF		LD (HL)，CF_H	；CF_H 送该指令
0356	3E01		LD A，01_H	
0358	32BA2F		LD (SIFLG)，A	；置SIFLG 标志
035B	18C0		JR CCS2B-$	；转 $031D_H$
035D	2ABC2F	CCS13	LD HL，($2FBC_H$)	
0360	C3D302	CCS13′	JP CCS1A0	
0363	2ABE2F	CCS14	LD HL，($2FBE_H$)	
0366	18F8		JR CCS13′-$	
0368	3AF02F	CCS9	LD A，(DIG4)	
036B	B7		OR A	
036C	2823		JR Z，CCS94-$	；是否输入四位数？否，转 0391_H
036E	CD8C06		CALL UFOR2	；得到断点地址在 HL 中
0371	CDA506		CALL UFOR3	；得到断点个数送 B

地址	目的码	标号	指令	注释
0374	280B		JR Z, CCS92-$	；无断点，转去在表内设置了第一个断点
0376	78		LD A, B	
0377	FE05		CP 05_H	
0379	CACB00		JP Z, RESTR1	；断点是否已设了五个？是转初始化
037C	CD5906	CCS91	CALL UIX2	
037F	20FB		JR NZ, CCS91-$	；IX+3，B-1，直到找到第一个自由空间
0381	3ADD2F	CCS92	LD A, (BFLG)	
0384	3C		INC A	；断点标志加1回送
0385	DD7400		LD(IX+00), H	
0388	DD7501		LD(IX+01), L	；断点地址送断点表
038B	32DD2F	CCS93	LD(BFLG), A	
038E	C3EB00		JP DISUP	；转显示
0391	CDA506	CCS94	CALL UFOR3	
0394	28F5		JR Z, CCS93-$	；是否设过断点？否，转$038B_H$
0396	DD7E02	CCS95	LD A, (IX+02)	；从断点表取操作码送 A
0399	DD6E01		LD L, (IX+01)	
039C	DD6600		LD H, (IX+00)	；从断点表取断点地址送 HL
039F	77		LD(HL), A	；操作码送回断点处
03A0	CD5906		CALL UIX2	
03A3	20F1		JR NZ, CCS95-$	；直到所有操作码都恢复完
03A5	AF		XOR A	；A 清另
03A6	18E3		JR CCS93-$	；转 $038B_H$
03A8	210010	CCS16：	LD HL, 1000_H	；EP ROM 编程首址1000→HL
03AB	010008		LD BC, 0800_H	；　　检查字节0800→BC
03AE	3EFF		LD A, FF_H	；　　检查内容FF→A

地址	目的码	标号	指令	注　释
03B0	EDA1	CCS16A：	CPI	
03B2	2006		JR NZ, CCS16B-$	；检查片子内容正确否？否，转03BA$_H$
03B4	EAB003		JP PE, CCS16A	；是否查完？否，转03B0$_H$继续查
03B7	C3CB00		JP RESTR1	；查完转回RESTR1
03BA	DD21F72F	CCS16B：	LD IX, DISMEM	
03BE	2B		DEC HL	
03BF	C3FE01		JP CCS5B	；显示出错地址
03C2	3E01	CCS20	LD A, 01$_H$	
03C4	32F22F		LD(PRFLG), A	；EPROM编程标志置位
03C7	CD8C06		CALL UFOR2	
03CA	E5		PUSH HL	
03CB	C1		POP BC	；得到字节数送BC
03CC	E5		PUSH HL	；保护字节数
03CD	CDA904		CALL CCS18K	；RAM源地址送HL，目的地址送DE
03D0	E5		PUSH HL	；存源地址
03D1	D5		PUSH DE	；存目的地址
03D2	3E25	CCS16C：	LD A, 25$_H$	
03D4	D386		OUT(CTC2), A	
03D6	3ECB		LDA, CB$_H$	
03D8	D386		OUT(CTC2), A	；CTC以定时方式工作，每隔26ms输出$\overline{ZC/TO2}$
03DA	3E80		LD A, 80$_H$	
03DC	D38C		OUT (DIGLH), A	；清显示，置PROM编程允许
03DE	EDA0		LDI	；插入等待状态，直到$\overline{PROMSEL}$有效
03E0	3E00		LD A, 00$_H$	
03E2	D38C		OUT(DIGLH), A	；清编程允许

地址	目的码	标号	指令	注　释
03E4	3E03		LD A, 03_H	
03E6	D386		OUT(CTC2), A	；CTC 复位
03E8	EAD203		JP PE, CCS16C	；如 BC－1≠0，循环返回
03EB	D1		POP DE	；恢复目的地址
03EC	E1		POP HL	；恢复源地址
03ED	C1		POP BC	；恢复字节数
03EE	1A	CCS16D：	LD A, (DE)	；取已写入 EPROM 数据
03EF	EDA1		CPI	；与原数据比较
03F1	2006		JR NZ, CCS16D-$	；有错，转$03F9_H$
03F3	E2CB00		JP PO, RESTR1	；无错返回
03F6	13		INC DE	；更新DE
03F7	18F5		JR CCS16D-$	；检查下一个字节
03F9	F5	CCS16D	PUSH AF	；保护错误数据
03FA	C5		PUSH BC	；保护字节计数
03FB	D5		PUSH DE	；保护错误地址
03FC	D9		EXX	；保护主寄存器
03FD	D1		POP DE	；DE 为 EPROM 出错地址
03FE	C1		POP BC	；BC为字节计数
03FF	C3F204		JP CCS17B	；转$04F2_H$，显示出错地址
0402	CD3A03	CCS18：	CALL CCS2D	；清显示
0405	32ED2F		LD(FLG24), A	；为发逻辑1而设置 FLG24
0408	ED5E		IM 2	；置中断方式 2
040A	01FA07		LD BC, CTC_1P	
040D	78		LD A, B	
040E	ED47		LD I, A	

地址	目的码	标号	指令	注释
0410	79		LD A, C	
0411	D384		OUT(CTC0), A	；中断向量表地址 $07FA_H$
0413	3E85		LD A, 85_H	
0415	D385		OUT(CTC1), A	
0417	3E1A		LD A, $1A_H$	
0419	D385		OUT(CTC), A	；CTC1 中断允许，每隔 208us 定时输出 1 正脉冲
041B	CDA904		CALL CCS18K	；起始地址→DE，结束地址→HL
041E	AF		XOR A	
041F	ED52		SBC HL, DE	
0421	23		INC HL	
0422	E5		PUSH HL	；转贮字节数送堆栈保护
0423	210000		LD HL, 0000_H	
0426	0603		LD B, 03_H	；设置计数值以获得40秒延时
0428	FB		EI	；CPU 允许中断
0429	76	CCS18A：	HALT	；等待CTC1 中断
042A	2D		DEC L	
042B	20FC		JR NZ, CCS18A-$	
042D	25		DEC H	
042E	20F9		JR NZ, CCS18A-$	
0430	10F7		DJNZ CCS18A-$	；延时循环，40秒全 1 开头段
0432	3E3A	CCS18B：	LD A, $3A_H$	
0434	CDFF06		CALL OTCHR1	；输出冒号
0437	AF		XOR A	；清进位位
0438	011000		LD BC, 0010_H	；标准块长度 0010_H→BC
043B	E1		POP HL	；恢复总转贮长度

地址	目的码	标号	指令	注　释
043C	ED42		SBC HL, BC	；HL 和最大字节数 16 相比
043E	3009		JR NC, CCS18C-$	；HL≥16，NC=1　转 0449H
0440	09		ADD HL, BC	；当少于16个字节时，恢复数据块字节数
0441	85		ADD A, L	；如L=0 将设置状态标志 Z
0442	47		LD B, A	；最后一块转贮长度存 B
0443	2E00		LD L, 00H	
0445	383C		JR Z, CCS18F-$	；Z=1 说明文件结束转 0483H
0447	1801		JR CCS18D-$ ；	；Z=0 块大小不为 0，转贮最后一个数据块
0449	41	CCS18C：	LD B, C	；数据块大小送 B
044A	E5	CCS18D：	PUSH HL	；存还需要转贮的长度
044B	0E00		LD C, 00H	；清检查和
044D	78		LD A, B	
044E	CDDD06		CALL UPACCS	；转贮记录长度
0451	3AC02F		LD A, (2FC0H)	
0454	67		LD H, A	
0455	CDDD06		CALL UPACCS	；转贮起始地址高字节
0458	3AC12F		LD A, (2FC1H)	
045B	6F		LD L, A	
045C	CDDD06		CALL UPACCS	；转贮起始地址低字节
045F	AF		XOR A	
0460	CDDD06		CALL UPACCS	；转贮“记录类型”。
0463	7E	CCS18E：	LD A, (HL)	
0464	CDDD06		CALL UPACCS	；转贮数据
0467	23		INC HL	；指针加 1
0468	10F9		DJNZ CCS18E-$	；直到 B=0

地址	目的码	标号	指令	注　释
046A	97		SUB A	；清累加器
046B	91		SUB C	；00-C=块校验和
046C	CDDD06		CALL UPACCS	；转贮"检查和"
046F	3E0D		LD A, $0D_H$	
0471	CDFF06		CALL OTCHR1	；转贮回车字符
0474	3E0A		LD A, $0A_H$	
0476	CDFF06		CALL OTCHR1	；转贮换行字符
0479	7C		LD A, H	
047A	32C02F		LD($2FC0_H$), A	；保存下一块转贮地址高字节
047D	7D		LD A, L	
047E	32C12F		LD($2FC1_H$), A	；保存下一块转贮地址低字节
0481	18AF		JR CCS18B-$	；继续转贮更多的数据
0483	0603	CCS18F:	LD B, 03_H	；设置计数值3，以获得转贮三个0
0485	AF	CCS18G:	XOR A	
0486	4F		LD C, A	
0487	CDDD06		CALL UPACCS	
048A	10F9		DJNZ CCS18G-$	；转贮三个0
048C	3E01		LD A, 01_H	
048E	CDDD06		CALL UPACCS	；转贮"记录类型"01
0491	97		SUB A	
0492	91		SUB C	
0493	CDDD06		CALL UPACCS	；转贮"文件结束"的检查和
0496	21FFFF	CCS18H:	LD HL, $FFFF_H$	；设置1秒全1结尾的计数值
0499	FB		EI	；开中断
049A	76	CCS18J:	HALT	；等待CTC1 中断

地址	目的码	标号	指令	注　释
049B	2D	CCS18J′	DEC L	
049C	20FD		JR NZ, CCS18J′-$	
049E	25		DEC H	
049F	20FA		JR NZ CCS18J′-$	；延时循环，1 秒全 1 结尾段
04A1	F3		DI	；关中断
04A2	3E03		LD A, 03_H	
04A4	D385		OUT(CTC1), A	；关 CTC
04A6	C3CB00		JP RESTR1	；转初始化
04A9	3AC02F	CCS18K：	LD A, ($2FC0_H$)	
04AC	57		LD D, A	；($2FC0_H$)→D
04AD	3AC12F		LD A, ($2FC1_H$)	；($2FC1_H$)→E
04B0	5F		LD E, A	
04B1	3AC22F		LD A, ($2FC2_H$)	
04B4	67		LD H, A	；($2FC2_H$)→H
04B5	3AC32F		LD A, ($2FC3_H$)	；($2FC3_H$)→L
04B8	6F		LD L, A	
04B9	C9		RET	；返回
04BA	ED5E	CCS17：	IM 2	；中断方式 2
04BC	21FF0F		LD HL, $0FFF_H$	；开始15秒延时
04BF	0620		LD B, 20_H	
04C1	2D	CCS17A：	DEC L	
04C2	20FD		JR NZ, CCS17A-$	
04C4	25		DEC H	
04C5	20FA		JR NZ, CCS17A-$	
04C7	10F8		DJNZ CCS17A-$	

地址	目的码	标号	指令	注　释
04C9	CD5A07	CCS17G：	CALL INCHR	；取得第一个字
04CC	D63A		SUB $3A_H$	；检测是否是冒号
04CE	20F9		JR NZ, CC17G-$	；不是冒号，循环返回
04D0	4F		LD C A	；"检查和"预置
04D1	CDE306		CALL ULACC	；收到记录长度及其检查和
04D4	47		LD B, A	
04D5	CDE306		CALL ULACC	；收到地址高字节
04D8	57		LD D, A	
04D9	CDE306		CALL ULA CC	；收到地址低字节
04DC	5F		LD E, A	
04DD	CDE306		CALL ULACC	；收到记录类型
04E0	3D		DEC A	；记录类型为"文件结束"吗？
04E1	F5		PUSH AF	
04E2	2807		JR Z, CCS17J-$	；如是则转移
04E4	CDE306	CCS17H：	CALL ULACC	；不是，接收数据
04E7	12		LD (DE), A	；存储数据
04E8	13		INC DE	
04E9	10F9		DJNZ CCS7H-$	；为接收更多的数据而循环
04EB	CDE306	CCS17J：	CALL ULACC	；接收"检查和"
04EE	AF		XOR A	；清累加器
04EF	81		ADD A, C	
04F0	2810		JR Z, CCS17K-$	；检查和校验无误转 0502_H
04F2	DD21F72F	CCS17B：	LD IX, DISMEM	；指向显示缓冲区
04F6	7A		LD A, D	
04F7	CD7906		CALL UFOR1	；显示高字节

地址	目的码	标号	指令	注　释
04FA	7B		LD A, E	
04FB	CD5106		CALL IXINC2	；显示低字节
04FE	F1		POP AF	；恢复堆栈指针 SP
04FF	C3EB00		JP DISUP	；转显示
0502	F1	CCS17K：	POP AF	；判断是否为"文件结束"
0503	20C4		JR NZ, CCS17G-$	；否，转入下一个记录的循环
0505	C3CB00		JP RESTR1	；重新启动监控程序
0508	DDE5	CCS21	PUSH IX	
050A	C1		POP BC	；目的地址送 BC
050B	FDE5		PUSH IY	
050D	E1		POP HL	；源地址送 HL
050E	23		INC HL	；指向偏移量
050F	E5		PUSH HL	
0510	D1		POP DE	；保护偏移量地址
0511	13		INC DE	；指向下一条指令的操作码
0512	79		LD A, C	；取低字节
0513	93		SUB E	；减
0514	5F		LD E, A	；保存偏移量
0515	77		LD (HL), A	；将偏移量填入需要的存储单元
0516	78		LD A, B	；取高字节
0517	9A		SBC A, D	；减
0518	DD21F72F		LD IX, DISMEM	
051C	CD7906		CALL UFOR1	；将高字节减法结果写入显示缓冲区
051F	7B		LD A, E	
0520	CDCC05		CALL ALTER1	；偏移量写入显示缓冲区

地址	目的码	标号	指令	注　释
0523	C3EB00		JP DISUP	；显示
0526	D9	CCS20D:	EXX	
0527	13		INC DE	
0528	C3ED03		JP CCS20B	
052B	CDA904		CALL CCS18K	；需清另的首址送 DE，末址 HL
052E	AF		XOR A	；A累加器清 0
052F	ED52		SBC HL，DE	
0531	E5		PUSH HL	
0532	C1		POP BC	
0533	03		INC BC	；计算字节数送 BC
0534	21E103		LD HL，03E1H	；HL 指向 $03E1_H$
0537	EDA0		LDI	；内存单元清另
0539	EA3405		JP PE，0534_H	；BC 是等于零？否，转 0534_H
053C	C3CB00		JP RESTR1	；是，显示提示符
053F	AF	CCS18K：	XOR A	；A清另
0540	F5	CCS18K1：	PUSH AF	；存 A
0541	CDA904		CALL CCS18K	；待检查的内存首地址送 DE，未地址送 HL
0544	F1		POP AF	；A＝0
0545	ED52		SBC HL，DE	
0547	23		INC HL	
0548	E5		PUSH HL	
0549	C1		POP BC	；字节数送 BC
054A	3810		JR C，CCS18K3-$	；首址小于末址，转 $055C_H$
054C	EB		EX DE，HL	；DE，HL 交换
054D	77	CCS18K2：	LD (HL)，A	；向内存单元送数

地址	目的码	标号	指令	注释
054E	F5		PUSH AF	；存 A
054F	AE		XOR (HL)	；从内存取数与 A 比较
0550	23		INC HL	；指向下一个内存
0551	200C		JR NZ, CCS18K4-$	；比较内容不一致，转 $055F_H$
0553	0B		DEC BC	；字节数减 1
0554	78		LD A, B	
0555	B1		OR C	
0556	200A		JR NZ, CCS18K5-$	；字节数 BC=0 吗？否，转 0562_H
0558	F1		POP AF	；恢复 A
0559	3C		INC A	；A 加 1
055A	20E4		JR NZ, CCS18K1-$	；A不为 0 转 0540_H
055C	C3CB00	CCS18K3：	JP RESTR1	；A 为 0 说明已查完，转 RESTR1
055F	C3BA03	CCS18K4：	JP CC16B	；出错处理
0562	F1	CCS18K5:	POP AF	；恢复 A
0563	18E8		JR CCS18K2-$	；转 $054D_H$
0565	DD21F72F	DIUPCA：	LD IX, DISMEM	；IX 指向显示缓冲区
0569	2AD92F		LD HL, (STKPT)	
056C	2B		DEC HL	
056D	7E		LD A, (HL)	
056E	CD7906		CALL UFOR1	
0571	2B		DEC HL	
0572	7E		LD A, (HL)	
0573	CD5106		CALL IXINC2	；用户栈中PC的内容取出送DSMEM0～DSEME3
0576	2B		DEC HL	
0577	7E		LD A, (HL)	

地址	目的码	标号	指令	注　释
0578	CD5106		CALL IXINC2	；A 的内容送 DSMEM4.5
057B	C9		RET	；返回
057C	21F72F	DISUP0：	LD HL，DISMEM	；HL = 2FF7H 显示缓冲区首址
057F	0620		LD B，20H	；B 指向扫描显示的最高位
0581	5E	DISUP1：	LD E，(HL)	；取字形的序号
0582	1600		LD D，00H	；D = 0
0584	3E00		LD A，00H	
0586	D38C		OUT (DIGLH)，A	；关显示
0588	DD21A607		LD IX，SEGPT	；变址寄存器指向字形表
058C	DD19		ADD IX，DE	；得到字形在表中的地址
058E	DD7E00		LD A，(IX + 0)	；从字形表得到字形的模
0591	D388		OUT (SEGLH)，A	；输出字形到字形选择锁存器
0593	78		LD A，B	
0594	D38C		OUT (DIGLH)，A	；输出到数位选择锁存器
0596	1E2D		LD E，2DH	
0598	1D	DISUP2：	DEC E	
0599	3E00		LD A，00H	
059B	BB		CP E	
059B	20FA		JR NZ，DISUP2－$	；延时1ms
059E	3E01		LD A，01H	
05A0	B8		CP B	
05A1	2805		JR Z，DISUP3－$	；是否已扫描了六个显示器？
05A3	23		INC HL	；否指向下一位
05A4	CB38		SRL B	；移位到下一位
05A6	18D9		JR DISUP1－$	；转0581H

地址	目的码	标号	指令	注　释
05A8	C9	DISUP3：	RET	；返回
05A9	DD21F72F	ALTER：	LD IX, DISMEM	；IX 指向存贮器显示缓冲区首址
05AD	3AF32F		LD A, (PFLG)	
05B0	B7		OR A	
05B1	2021		JR NZ, ALTER2-$	；口内容的改变转 $05D4_H$
05B3	3AF52F		LD A, (RFLG)	
05B6	B7		OR A	
05B7	2026		JR NZ, ALTER3-$	；寄存器内容的改变转 $05DF_H$
05B9	3AF62F		LD A, (MFLG)	
05BC	B7		OR A	
05BD	205E		JR NZ, ALTER4-$	；辅助寄存器内容的改变转 $061D_H$
05BF	3AF42F		LD A, (MFLG)	
05C2	B7		OR A	
05C3	2854		JRZ, ALTE3C-$	；不正确操作，转入 RESTR
05C5	CD8C06		CALL UFOR2	；取得地址送 HL
05C8	CD4706	ALTERA：	CALL ALTER7	
05CB	77		LD (HL), A	；从DSMEM6.7取修改数送HL
05CC	DD21FB2F	ALTER1：	LD IX, DSMEM4	
05D0	CD7906		CALL UFOR1	
05D3	C9		RET	；把新的数据写入显示缓冲区 DSMEM4.5
05D4	CD8C09	ALTER2：	CALL UFOR2	；取得地址送 HL
05D7	CD4706		CALL ALTER7	；从 DSMEM6.7 取修改数送HL
05DA	4C		LD C, H	
05DB	ED79		OUT(C), A	；新数据写入口中
05DD	18ED		JR ALTER1-$	；转 $05CC_H$

地址	目的码	标号	指令	注　释
05DF	21D507	ALTER3：	LD HL，REGTB	；寄存器表地址送 $07D5_H$
05E2	CD2506		CALL ALTER5	；找R在用户堆栈中位置送 HL，IX＝DSMEM
05E5	7B		LD A，E	；键号送 A
05E6	FE06		CP 06_H	；与 6 比较置 C
05E8	3802		JR C，ALTE3A－$	；C＝1，R＜6，转 $05EC_H$
05EA	18DC		JR ALTERA－$	；C＝0为8bit 寄存器转 $05CB_H$
05EC	FE03	ACTE3A：	CP 03_H	
05EE	281F		JR Z，ALTE3B－$	；是键 IFF 吗？是，转 $060F_H$
05F0	FE02		CP 02_H	
05F2	2825		JR Z，ALTE3C－$	；是键SP吗？是转0619_H
05F4	DD21F52F		LD IX，$2FF5_H$	
05F8	CD4706		CALL ALTER7	；从 DSMEM4.5 取 rP_H
05FB	23		INC HL	
05FC	77		LD (HL)，A	；修改高字节
05FD	DD21F92F		LD IX，DSMEM2	
0601	CD7906		CALL UFOR1	；rp_H→DSMEM2.3
0604	DD21F72F		LD IX，DISMEM	
0608	CD4706		CALL ALTER7	；从 DSMEM6.7 取 rp_L
060B	2B		DEC HL	
060C	77		LD (HL)，A	；修改低字节
060D	18BD		JR ALTER1－$	；转 $05CC_H$
060F	3AFE2F	ALTE3B：	LD A，(DSMEM7)	；获得新值(04或00)
0612	32EE2F		LD (UIF)，A	；修改 UIF
0615	32FC2F		LD (DSMEM5)，A	；写入 DSMEM5
0618	C9		RET	；返回

地址	目的码	标号	指令	注　释
0619	E1	ALTE3C:	POP HL	；堆栈平衡
061A	C3D000		JP RESTR3	；转RESTR1
061D	21E507	ALTER4:	LD HL, REGTB′	；辅助寄存器表首址送 HL
0620	CD2506		CALL ALTER5	；找到R′在堆栈位置送HL
0623	18A3		JR ALTERA-$	；转$05C8_H$
0625	DD21F72F	ALTER5:	LD IX, DISMEM	；IX 指向 DSMEM 其中存有寄存器代号
0629	DD5E00		LD E, (IX+0)	；键号送E
062C	0600		LD B, 00_H	
062E	50		LD D, B	；D, B清另
062F	19		ADD HL, DE	；表地址＝首址＋键号
0630	4E		LD C, (HL)	
0631	79		LD A, C	；取序号与19比较(序号—寄存器在用户堆栈中的地址与用户堆栈底的差值)
0632	FE19		CP 19_H	
0634	CAD000		JP Z, RESTR3	；非法值转 RESTR1
0637	2AD92F		LD HL, (STKPT)	；从 STKPT 取用户 SP
063A	B7		OR A	；C＝0
063B	ED42		SBC HL, BC	；从 SP 减去偏移找到 R 在映象区地址
063D	C9		RET	；返回
063E	CB27	ALTER6:	SLA A	
0640	CB27		SLA A	
0642	CB27		SLA A	
0644	CB27		SLA A	；左移四移
0646	C9		RET	
0647	DD7E06	ALTER7:	LD A, (IX+06)	；将显示缓冲区内容相或形成一个完整字节
064A	CD3E06		CALL ALTER6	

地址	目的码	标号	指令	注　释
064D	DDB607		OR (IX+07)	;
0650	C9		RET	
0651	DD23	IXINC2	INC IX	
0653	DD23		INC IX	; IX=IX+2
0655	CD7906		CALL UFOR1	; 将A写入(IX+0)(IX+1)
0658	C9		RET	
0659	DD23	UIX2:	INC IX	
065B	DD23		INC IX	
065D	DD23		INC IX	
065F	05		DEC B	
0660	C9		RET	; IX=IX+3　　B=B-1
0661	21FF08	D20MS:	LD HL, $08FF_H$	
0664	2D	D20MS1:	DEC L	
0615	20FD		JR NZ, D20MS1-$	
0667	25		DEC H	
0668	20FA		JR NZ, D20MS1-$	
066A	C9		RET	; 延时 20ms
066B	21F72F	UFGCR:	LD HL, DISMEM	
066E	22DB2F		LD(KEYPTR), HL	; 显示指针指向 $2FF7_H$
0671	AF		XOR A	; A 清另
0672	0608		LD B, 08_H	
0674	2B	UFGCR1:	DEC HL	
0675	77		LD (HL), A	
0676	10FC		DJNZ UFGCR1-$	; 清各状态标志
0678	C9		RET	; 返回

地址	目的码	标号	指令	注　释
0679	47	UFOR1：	LD B，A	；保护A
067A	E60F		AND $0F_H$	；屏蔽掉字节的高四位
067C	DD7701		LD (IX+1)，A	；将低四位写入显示缓冲区(IX+1)
067F	78		LD A，B	；取回完整字节
0680	CB3F		SRL A	
0682	CB3F		SRL A	
0684	CB3F		SRL A	
0686	CB3F		SRL A	；右移四位
0688	DD7700		LD (IX+0)，A	；写入显示缓冲区(IX+0)
068B	C9		RET	；返回
068C	DD21F72F	UFOR2：	LD IX，DISMEM	；IX 指向 DISMEM
0691	DD7E00		LD A，(IX+0)	；取第一个数高四位送 A
0693	CD3E06		CALL ALTER6	；A 左移四位
0696	DDB601		OR(IX+1)	；与低四位和或得到地址的高字节
0699	67		LD H，A	；存入 H
069A	DD7E02		LD A，(IX+2)	；取第二字节的高四位
069D	CD3E06		CALL ALTER6	；左移四位
06A0	DDB603		OR (IX+03)	；与低四位和或，得到地址的低字节
06A3	6F		LD L，A	；存入L
06A4	C9		RET	；返回
06A5	DD21DE2F	UFOR3：	LD IX，BPTAB	；IX 指向断点表的首地址
06A9	3ADD2F		LD A，(BFLG)	
06AC	B7		OR A	
06AD	47		LD B，A	；断点状态数送B，B=0，Z=1
06AE	C9		RET	；返回

地址	目的码	标号	指令	注　释
06AF	0608	UFOR4：	LD B，08_H	
06B1	21F72F		LD HL，DISMEM	
06B4	3E10		LD A，10_H	
06B6	77	UFOR4A：	LD (HL)，A	
06B7	23		INC HL	
06B8	10FC		DJNZ UFOR4A-$	
06BA	C9		RET	；熄灭符序号 10_H 送显示缓冲区
06BB	D630	UABIN：	SUB 30_H	
06BD	FE0A		CP $0A_H$	
06BF	F8		RET M	
06C0	D607		SUB 07_H	
06C2	C9		RET	；将累加器中的一个 ASCII 码转换为二进制
06C3	E60F	UBASC：	AND $0F_H$	
06C5	C690		ADD A，90_H	
06C7	27		DAA	
06C8	CE40		ADC A，40_H	
06CA	27		DAA	
06CB	C9		RET	；将累加器A中的一个4位二进制数转换为ASCII码
06CC	D9	UPACC：	EXX	；OTCHR 使用辅助寄存器
06CD	F5		PUSH AF	；保护 A 以输入第二个 ASCII 字符
06CE	0F		RRC A	
06CF	0F		RRC A	
06D0	0F		RRC A	
06D1	0F		RRC A	；交换A累加器高、低字节的位置
06D2	CDFC06		CALL OTCHR	；转贮输出原 A 累加器中高四位

地址	目的码	标号	指令	注　释
06D5	F1		POP AF	；恢复 A
06D6	E60F		AND $0F_H$	；屏蔽掉高四位
06D8	CDFC06		CALL OTCHR	；转贮输出原A累加器中低四位
06DB	D9		EXX	；恢复寄存器
06DC	C9		RET	；返回
06DD	F5	UPACCS：	PUSH AF	；保护累加器内容—要转贮的二进制数
06DE	81		ADD A, C	
06DF	4F		LD C, A	；求检查和后送入 C 寄存器
06E0	F1		POP AF	；恢复 A
06E1	18E9		JR UPACC-$	；累加器内容转贮
06E3	C5	ULACC：	PUSH BC	；保护 C 寄存器(检查和)
06E4	CD5A07		CALL INCHR	；读第一个 ASCII 字符
06E7	CDBB06		CALL UABIN	；转换为二进制
06EA	07		RLC A	
06EB	07		RLC A	
06EC	07		RLC A	
06ED	07		RLC A	；移到高四位
06EE	4F		LD C, A	；保留高四位于 C
03EF	CD5A07		CALL INCHR	；取第2个 ASCII 字符
06F2	CDBB06		CALL UABIN	；转换为 2 进制
06F5	B1		OR C	；高，低位相或形成完整的二进制数
03F6	C1		POP BC	；取回检查和
06F7	F5		PUSH AF	；保护累加器内容
03F8	81		ADD A, C	
06F9	4F		LD C, A	

地址	目的码	标号	指令	注释
06FA	F1		POP AF	
06FB	C9		RET	
06FC	CDC306	OTCHR:	CALL UBASC	；二进制转 ASCII 码存 A
06FF	FB	OTCHR1:	EI	；开中
0700	57		LD D, A	；被转贮的 ASCII 码送 D 寄存器
0701	3E10		LD A, 10_H	；为输出停止位将周期计数值定为16
0703	2E0A		LD L, $0A_H$	；ASCII 标准字节长度为 10 送 L 计数器
0705	CB12		RL D	；CY→D
0707	FE01	OTCHR2:	CP 01_H	
0709	20FC		JR NZ, OTCHR2-$	；A=1? 等待到 A=1 为止　即输出15 个 4800Hz 脉冲
070B	47		LD B, A	；保存累加器内容 A=1
070C	AF		XOR A	
070D	32ED2F		LD (FLG24), A	；为发出起始位而清标志(FLG24)
0710	78		LD A, B	；A=1
0711	76	OTCHR3:	HALT	；停机，直到发最后一个脉冲
0712	37		SCF	；设置停止位(最后)CY=1
0713	CB1A		RR D	；带进位右移 D一次，CY→D7 即先输出低位
0715	2D		DEC L	；L≠0　Z=0　NZ=1 转 $071F_H$
0716	2007		JR NZ, OTCHR4-$	
0718	3E01		LD A, 01_H	
071A	32ED2F		LD (FLG24), A	；字发完——跟随发1
071D	F3		DI	；关中
071E	C9		RET	；返回
071F	FE01	OTCHR4:	CP 01_H	；反复等待下一次计数到0，直到最后一次A=1为止

地址	目的码	标号	指令	注　释
0721	20FC		JR NZ, OTCHR4-$	；输出 7 个 2400Hz 或 15 个 4800Hz 脉冲
0723	CB42		BIT 0, D	
0725	2008		JR NZ, OTCHR5-$	；测试下一位
0727	47		LD B, A	
0728	AF		XOR A	
0729	32ED2F		LD (FLG24), A	
072C	78		LD A, B	
072D	18E2		JR OTCHR3-$	；下一位是0将0送入累加器，清FLG24 标志 A=1
072F	47	OTCHR5：	LD B, A	
0730	3E01		LD A, 01_H	
0732	32ED2F		LD (FLG24), A	
0735	78		LD A, B	
0736	18D9		JR OTCHR3-$	；下一位是 1，将 1 送入累加器，置 FLG24 标志
0738	3D	OTCHR6：	DEC A	；A 为周期计数器
0739	201C		JR NZ, OTCHR8-$	；计数未完，转中断返回
073B	3E85		LD A, 85_H	
073D	D385		OUT(CTC1), A	；CTC 允许中断，以定时方式工作
073F	DD21ED2F		LD IX, FLG24	
0743	DDCB0046		BIT 0, (IX+0)	
0747	2008		JR NZ, OTCHR7-$	；测试(FLG24)标志不为 0 转 0751_H
0749	3E34		LD A, 34_H	
074B	D385		OUT(CTC1), A	；CTC 设为 2400Hz，每隔 416us 发一脉冲
074D	3E08		LD A, 08_H	；周期计数为 8
074F	1806		JR OTCHR8-$	；转 0757_H 返回
0751	3E1A	OTCHR7：	LD A, $1A_H$	

地址	目的码	标号	指令	注　释
0753	D385		OUT(CTC_1), A	；CTC 设为 4800Hz，每隔 208us 发一次脉冲
0755	3E10		LD A, 10_H	；计数周期为16
0757	FB	OTCHR8：	EI	；CPU 开中，保证后继中断被响应
0758	ED4D		RETI	；返回
075A	21FE07	1NCER	LD HL, CTC_3L	
075D	7C		LD A, H	
075E	ED47		LD I, A	
0760	7D		LD A, L	
0761	D384		OUT(CTC_0), A	；中断向量表地址为 $07FE_H$
0763	0608	1NCHR1：	LD B, 08_H	；一个字节的位计数值08送 B
0765	FB		EI	；开中
0766	AF		XOR A	；A 清另
0767	67		LD H, A	；H清另
0768	DB90	1NCHR1A：	IN A, (KBSEL)	
076A	CB7F		BIT 7, A	
076C	20FA		JR NZ, INCHR1A-$	；为找起始位‘0’而循环返回
076E	3EA5		LD A, $0A5_H$	
0770	D387		OUT(CTC_3), A	；为了确保数据读入正确在中点进行测试，故设置
0772	3E0D		LD A, $0D_H$	CTC 以定时方式工作，隔 1.66ms 发一个脉冲
0774	D387		OUT(CTC_3), A	(数据传送字 300bit/秒　(bit = 3.33ms)
0776	3EA5		LD A, $A5_H$	
0778	D387		OUT(CTC_3), A	
077A	3E1A		LD A, $1A_H$	
077C	D387		OUT(CTC_3), A	；以后 CTC 每隔 3.33ms 发一个脉冲测试信号
077E	76		HALT	；等待中断

地址	目的码	标号	指令	注　释
077F	CB7F		BIT 7, A	
0781	2014		JR NZ, INCHR3-$	；再读一次，若是假启始位，则重新开始等待字符的第一位
0783	76	INCHR2：	HALT	
0784	E680		AND 80_H	
0786	B4		OR H	
0787	67		LD H, A	；七位 ASCII 码屏蔽掉其它位输入，送 H
0788	1018		DJNZ INCHR5-$	；是否已输入完一个字节，否转 $07A2_H$
078A	CB7F		BIT 7, A	；最高位因是“1” $Z \leftarrow \overline{A7}=0$ 若不是则说明格式错转 0797H 重新启动
078C	2809		JR Z, INCHR3-$	
078E	F3		DI	；关中
078F	3E03		LD A, 03_H	
0791	D387		OUT(CTC3), A	；CTC 复位
0793	7C		LD A, H	；将装入的 ASCII 字符送 A
0794	E67F		AND $7F_H$	；屏蔽掉停止位
0796	C9		RET	；返回
0797	3E03	INCHR3：	LD A, 03_H	
0799	D387		OUT(CTC3), A	；CTC3 复位，格式错误
079B	18C6		JR INCHR1-$	；为接收另外的字符循环返回
079D	DB90	INCHR4：	IN A, (KBSEL)	；输入数据
079F	FB		EI	；为后继中断产生开中
07A0	ED4D		RETI	；返回
07A2	CB0C	INCHR5：	RRC H	
07A4	18DD		JR INCHR2-$	；等待下一个
07A6	40	SEGPT：		；七段显示字形表　0
07A7	79			；　　　　　　　　1

地址	目的码	标号	指令	注　释
07A8	24			; 2
07A9	30			; 3
07AA	19			; 4
07AB	12			; 5
07AC	02			; 6
07AD	78			; 7
07AE	00			; 8
07AF	18			; 9
07B0	08			; A
07B1	03			; B
07B2	46			; C
07B3	21			; D
07B4	06			; E
07B5	0E			; F
07B6	7F			; 空
07B7	3F			; 提示符"－"
07B8	7D			; 撇号标记"/"
07B9	FF	KYTBL		; 键值表 0
07BA	EF			; 1
07BB	F7			; 2
07BC	FB			; 3
07BD	DF			; 4
07BE	E7			; 5
07BF	EB			; 6
07C0	CF			; 7

地址	目的码	标号	指令	注释
07C1	D7			; 8
07C2	DB			; 9
07C3	DD			; A
07C4	ED			; B
07C5	FD			; C
07C6	0D			; D
07C7	0B			; E
07C8	07			; F
07C9	0E			; EXEC
07CA	FE			; SS
07CB	EE			; MON
07CC	DE			; MON′
07CD	CD			; IFBC/NEXT
07CE	CB			; 2FBE/LAST
07CF	C7			; REG′/REG
07D0	BF			; TROM/MEM
07D1	BD			; LOAD/BP
07D2	BB			; DUMP/PORT
07D3	B7			; DELETE/INSERT
07D4	AF			; PROM/SI
07D5	19	REGTB		; 寄存器表
07D6	02			; 键1 = PC
07D7	02			; 键2 = SP
07D8	0C			; 键3 = IFF　显示用户IFF2
07D9	16			; 键4 = IX

地址	目的码	标号	指令	注　释
07DA	18			; 键5 = IY
07DB	0B			; 键6 = I
07DC	09			; 键7 = H
07DD	0A			; 键8 = L
07DE	19			; 键9 = 不显示
07DF	03			; 键A = A
07E0	05			; 键B = B
07E1	06			; 键C = C
07E2	07			; 键D = D
07E3	08			; 键E = E
07E4	04			; 键F = F
07E5	19	REGTB′		; 辅助寄存器　键 0 不显示
07E6	19			; "1"
07E7	19			; "2"
07E8	19			; "3"
07E9	19			; "4"
07EA	19			; "5"
07EB	19			; "6"
07EC	13			; 键 7 = H
07ED	14			; 键 8 = L
07EE	19			; 键 9 不显示
07EF	0D			; 键 A = A
07F0	0F			; 键 B = B
07F1	10			; 键 C = C
07F2	11			; 键 D = D

地址	目的码	标号	指令	注　释
07F3	12			;　　键E＝E
07F4	0E			;　　键F＝F
07F5	FF			
07F6	FF			
07F7	FF			
07F8	D6			
07F9	2F			;　CTC0 中断矢量表
07FA	38			
07FB	07			;　CTC1P 转贮中断矢量
07FC	FF			
07FD	FF			
07FE	9D			
07FF	07			;　CTC3L 磁带装入中断矢量

AM 区域工作单元分配表

地址	标号	注　释
2FBA	SIFLG	;指令单处理标志
2FBB	OPSAVE	;操作码保存单元
2FBC		;为处理用户自定义处理程序
2FBD		插入入口地址
2FBE		
2FBF		
2FC0	PUNHSH	;转储起始地址高字节
2FC1	PUNHSL	;转储起始地址低字节
2FC2	PUNHEH	;转储结束地址高字节
2FC3	PUNHEL	;转储结束地址低字节

```
2FC4    RST16       ; 为处理RST16～56用户插入
2FC5                ; 转移指令
2FC6
2FC7    RST24
2FC8
2FC9
2FCA    RST32
2FCB
2FCC
2FCD    RST40
2FCE
2FCF
2FD0    RST48
2FD1
2FD2
2FD3    RST56
2FD4
2FD5
2FD6    CTC0V       ;   为CTC0 中断用户插入转移指令
2FD7
2FD8
2FD9    STKPT       ;   用户栈指针低字节
2FDA    STKPT1      ; 用户栈指针高字节
2FDB    KEYPTR      ; 为了写入下一个显示缓冲区单元而
2FDC                  设置的指针
2FDD    BFLG        ; 已设置的断点数目
```

地址	标号	注　释
2FDE	BPTAB	；断点地址转移来的操作码
2FED	FLG24	；转储信息标志
2FEE	UIF	；用户IFF2
2FEF	DIG2	；已输入 2 位数标志
2FF0	DIG4	；已输入 4 位数标志
2FF1	SSFLG	；单步方式标志
2FF2	PRFLG	；EPROM编程标志
2FF3	PFLG	；口检查标志
2FF4	MFLG	；存储器检查标志
2FF5	RFLG	；寄存器检查标志
2FF6	ARFLG	；辅助寄存器检查标志
2FF7	DISMEM	；存储器显示缓冲区
2FF8	DSMEM1	
2FF9	DSMEM2	
2FFA	DSMEM3	
2FFC	DSMEM5	
2FFD	DSMEMEM6	
2FFE	DSMEM7	
2FFF	SMON	；换档标志

CJ801

Z80单板计算机基础知识

目　　录

第一章　基 础 知 识

1.1　数制及不同数制间的换算

一、十进制数

在日常生活中，人们最熟悉的是十进制数。它有十个不同的数字：0，1，2，3，4,5，6，7，8，9。在表示数时，这些处于不同的位置(或数位)的数字代表的意义是不同的。例如1001，表示一千零一。我们称这是一个四位(十进制)数。

一般地讲，任何十进制数

$$D3\ D2\ D1\ D0$$

都可以写成基数十的各次幂的和式，即

$$D3\ D2\ D1\ D0 = D3 \times 10^3 + D2 \times 10^2 + D1 \times 10^1 + D0 \times 10^0$$

可见同样一个数字，放在最高位与最低位的含义是不同的，D3 有表示 10^3 的权，D0 有表示 10^0 的权。上式我们又称为按权展开式。

二、二进制数

在电子计算机中通常并不采用十进制数，而是采用二进制数。因为电子计算机中使用高电平和低电平来表示两个不同的数码：0，1。一个二进制数的按权展开式如下：

$$B3\ B2\ B1\ B0 = B3 \times 2^3 + B2 \times 2^2 + B1 \times 2^1 + B0 \times 2^0$$

在这种数制中，1001 不是表示一千零一，而是表示九。

三、十六进制数

这种数制中有十六个不同的数字：0，1，2，3，4，5，6，7，8，9，A(相应于十进制数中的10，B(11)，C(12)，D(13)，E(14)，F(15)。

它的按权展开式如下：

$$H3\ H2\ H1\ H0 = H3 \times 16^3 + H2 \times 16^2 + H1 \times 16^1 + H0 \times 16^0$$

在微型计算机中经常使用这种十六进制数制，其理由如下：

1．它与二进制数之间的转换比较方便。例如二进制数

$$1001\ 1100B^* = (1 \times 2^3 + 1 \times 2^0) \times 16^1 + (1 \times 2^3 + 1 \times 2^2) \times 16^0 = 9C_H$$

即每四个二进制数对应于一个十六进制数。微型机中的二进制数通常是 8 位或 8 位的整倍数(16,24,32位)，则相应的十六进制数为 2，4，6，8 位。

B (Binary)表示二进制数。同样 D(Decimal)和 H(Hexabccimal)分别表示十进制数和十六进制。

2. 使用二进制，书写太长，不易记忆，且念起来不易懂。而使用十六进制可以弥补上述缺点。

四、八进制数

它的特性与16进制数相似，并且在近代微型机中很少使用，故不予介绍。

五、二—十进制数(BCD码)

在这种数制中，用二进制数来表示十进制数字。例如

$$97D = 10010111\ BCD$$

但须注意，这种数制与二进制不同，它们的数位的权不同。

$$10010111 = (1\times2^3+1\times2^0)\times10^1+(1\times2^2+1\times2^1+1\times2^0)\times10^0$$

这种数制的优点是既照顾了人们使用十进制数的习惯，又考虑到计算机的特点。缺点是不便于运算。由于微型计算机中的运算比较简单，因此经常采用这种数制，特别是微型机化仪器的输入和输出。

六、不同数制间的换算

1. 二翻十。即把二进制数换算成十进制数。这种方法比较简单，利用二进制数的按权展开式，将二进制数按权相加，就得到等值的十进制数。

例：　$1101\,B = 1\times2^3+1\times2^2+0\times2^1+1\times2^0 = 13D$

2. 十翻二。即把十进制数换算成二进制数。现举例说明其方法：

例：　求13D的二进制数

$13\div2=6$	余数1	(最低位)
$6\div2=3$	余数0	
$3\div2=1$	余数1	
$1\div2=0$	余数1	(最高位)

结果：1101 B

在微型计算机中常有一些实用的子程序用来进行这类运算。

1.2 逻辑电路

通常一个逻辑电路有一或两个以上的输入，一个输出。不同性质的逻辑电路，输出与输入之间有不同的关系，这种关系称为逻辑函数。输入或输出的状态只有两种：高电平和低电平。通常，我们用逻辑1(即有效)来表示高电平，逻辑0(即无效)表示低电平，这称为正逻辑，大多数微型机均使用正逻辑。反之，用逻辑1表示低电平，逻辑0表示高电平，称为负逻辑，少数微型机如Intel4004/4040使用负逻辑。本书中使用正逻辑。

下面介绍通常用的逻辑电路。

一、与门(AND gate)

1. 符号：如图1.1(a)所示。

2. 含义：只有当所有输入 A 和 B 都为 1 状态时，输出 y 才为 1。

3. 逻辑式：$y = A \times B$ 或 $y = AB$。

4. 真值表(逻辑函数的另一种表示法)：如表 1.1所 示。

二、或门(OR gate)

1. 符号：如图 1.1(b)所示。
2. 含义：只要输入中有一个为逻辑 1，输出即为逻辑 1。
3. 逻辑式：$y = A + B$
4. 真值表：如表 1.1 所示

表 1.1 常用逻辑电路真值表

输入		输出				
A	B	与门	或门	非门 ($Y = \overline{A}$)	异门	同门
0	0	0	0	1	0	1
0	1	0	1	1	1	0
1	0	0	1	0	1	0
1	1	1	1	0	0	1

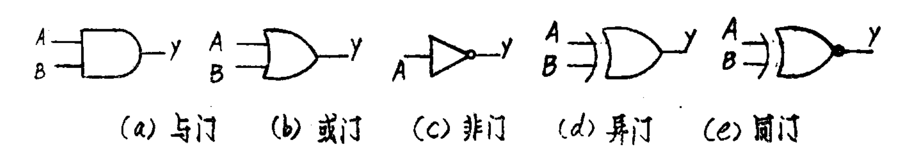


图 1.1 常用逻辑电路

三、非门(反相器—inverter)

1. 符号：如图 1.1(c)所示。
2. 含义：输出与输入状态相反。
3. 逻辑式：$y = \overline{A}$
4. 真值表：如表 1.1 表示。

四、异门(exclusive—OR gate)

1. 符号：如图 1.1(d)所示。
2. 含义：输入信号中有奇数个 1，输出为逻辑 1。反之，为逻辑 0。
3. 逻辑式：$y = A \oplus B$
4. 真值表：如表 1.1 所示。

五、同门(eXclusive—NOR gate)

1. 符号：如图 1.1(e)所示。
2. 含义：输入信号中有偶数个 1，输出为逻辑 1。反之，为逻辑 0。

逻辑式：$y = A \overline{\oplus} B$ 或 $y = \overline{A \oplus B}$

4. 真值表：如表1.1所示。

六、摩根定理

摩根定理是逻辑运算中最常用的定理。即

$$\overline{AB}=\overline{A}+\overline{B}$$

$$\overline{A+B}=\overline{A}\cdot\overline{B}$$

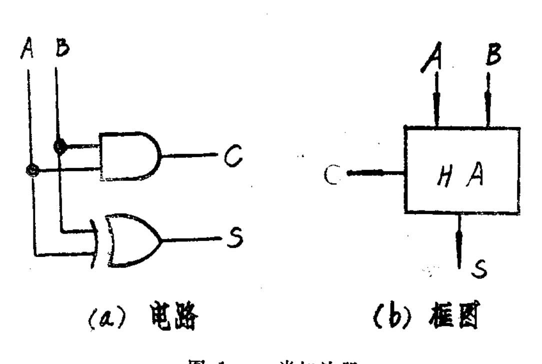
读者可以用真值表来证明这个定理。此定理表明，可以把逻辑“与”运算转换成逻辑“或”运算。反之亦然。在工程上，可以使用单一的与非门来实现各种逻辑运算。

1.3 运算电路(Arithmetic circuit)

一、一位(二进制)数加法与半加法器(Half adder)

两个一位的二进制数A与B相加，有四种情况：

	(1)	(2)	(3)	(4)
A	0	0	1	1
+B	+ 0	+ 1	+ 0	+ 1
C S	0 0	0 1	0 1	1 0
	↓ ↓	↓ ↓	↓ ↓	↓ ↓
	C S	C S	C S	C S



(a) 电路　　(b) 框图

图 1.　半加法器

其结果如上，其中S是本位和，C是(对下一位的)进位。由上节可知：

$$S=A\oplus B$$

$$C=A\times B$$

由此可得逻辑图，如图1.2(a)所示。其方框图如图1.2(b)所示，通常叫做半加法器。

二、多位(二进制)数加法与全加法器(Full adder)

先介绍多位数相加的过程。设有两个4位的二进制数A和B，A=1011，B=1001，试求A+B?

Ci	1←	0←		1←		1←		
Ai		1		0		1		1
+Bi		1		0		0		1
	1		0		1		1	
Si	↑	0	↑	1	↑	0	↑	0
	C_4	↑	C_3	↑	C_2	↑	C_1	↑
		S_3		S_2		S_1		S_6

下面按照小学生的习惯进行演算，即逐位高行运算。先从最低位开始，$A_0+B_0=$

C_1S_0，S_0 是本位和，C_1 是对下一位的进位。其次 $C_1+A_1+B_1=C_2S_1$，依此类推。

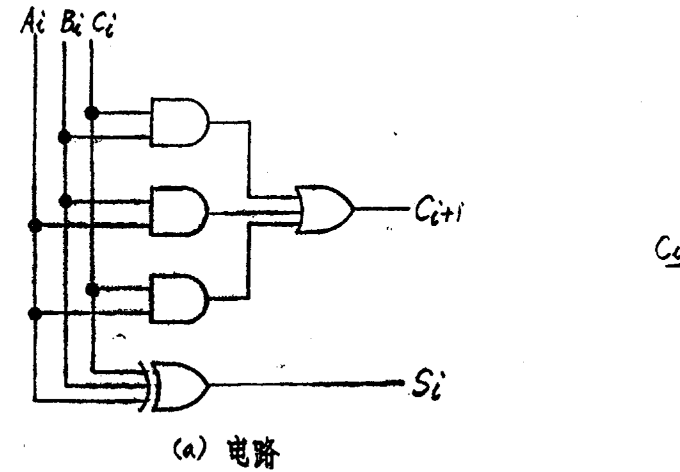
可见，多位数相加与一位数相加的差别在于：前者为三个一位数相加，后者只有两个一位数相加。三个一位数相加有八种情况，其结果表 1.2 所示。其逻辑式如下：

$$S_i=A_i\oplus B_i\oplus C_i$$

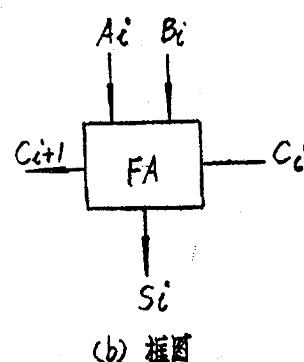
$$C_{i+1}=A_iB_i+B_iC_i+C_iA_i$$

表 1.2 全加法器真值表

输	入		输	出
Ai	Bi	Ci	C_{i+1}	Si
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



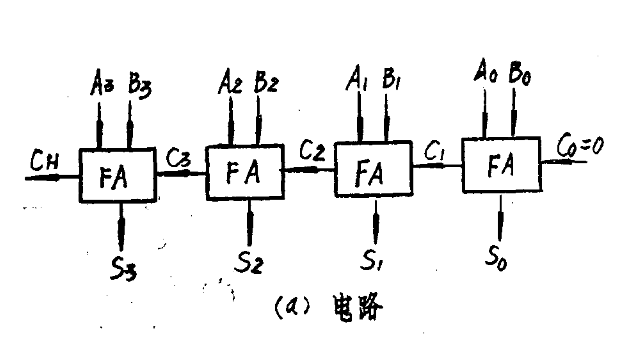
(a) 电路



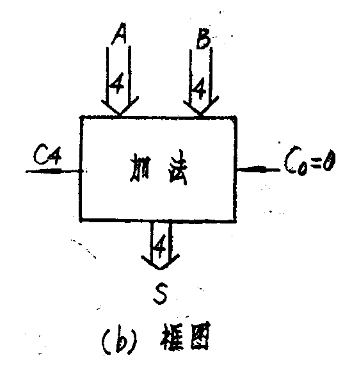
(b) 框图

图 1.3 全加法器

图1.3(a)示出其逻辑图，图 1.3(b)示出其方框图，通常叫做全加法器。由此可以得到两个 4 位数相加的二进制加法器，如图1.4(a)所示。将此图简化可得图1.4(b)所示的方框图。



(a) 电路



(b) 框图

图 1.4 二进制加法器

三、二进制数减法与补码运算

按照小学生的演算法则进行二进制数减去并不困难，但是用电子线路来实现减法要比加法麻烦得多。因此人们搞出一种新的表示数的形式——补码。

如前所述，四位二进制数字可以表示 16 个十进制数—0 到 15。现在重新规定其含义，用来表示另外 16 个数，如表 1.3 所示。0 到 7 八个数的表示法与以前相同；所不同的是它能表示负数。以-1 为例，表中用 1111 个来表示。数 1111 加 1 到 10000，最高（第五）位 1 将被丢失而不起作用。所以，数 1111 补上一个 1 得 0000，即数 1111 对 0000 来说缺 1，因而可将 1111 看作-1，其他类推。在补码表示法中，最高位可以看作符号位，0 表示正，1 表示负。由此，利用补码进行减法运算可以将减法转换为加法。例如

$5-3=5+(-3)=?$ 其运算过程如下：

$$\begin{array}{r} 5 \\ +(-3) \\ \hline 2 \end{array} \qquad \begin{array}{r} 0101 \\ +1101 \\ \hline 1\ 0010 \end{array}$$

（最高位 1：丢失）

表 1.3　数的补码表示

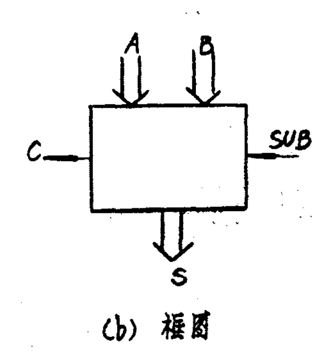
数	补　码	数	补　码
-8	1000	0	0000
-7	1001	1	0001
-6	1010	2	0010
-5	1011	3	0011
-4	1100	4	0100
-3	1101	5	0101
-2	1101	6	0110
-1	1111	7	0111

但是如何将 3(0011)转换成 -3(1101)？简单地说，就是将 0011 进行“求反加1”运算：

求反

$$\begin{array}{cccc} 0 & 0 & 1 & 1 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 1 & 1 & 0 & 0 \end{array}$$

(a) 电路



(b) 框图

图 1.5　加法/减法器

加

$$
\begin{array}{r}
\dot{1}\,1\,\dot{0}\,\dot{0} \\
+\quad\quad 1 \\
\hline
1\,1\,0\,1
\end{array}
$$

在电路上又如何实现？图1.5(a)示出了利用补码运算的加法/减法器。当进行加法运算时，令SUB＝0，则 $C_0=0$，且 $B_i'=B_i(i=0-3)$， 它与图1.4的二进制加法器相同。当进行减法运算时，令SUB＝1，则 $C_0=1$，且 $B_i'=\overline{B_i}(i=0-3)$。即通过异门将减数B求反，利用 $C_0=1$，得到加1操作。图1.5(b)示出其方框图。

1.4 触发器与存储器(Flip-flop and Register)

一、触发器

触发器是一种常用的数字电路。它可以作为一种无触点开关，也可以存放在一个(二进制)位(bit)的信息。常用的触发器有：

1. D触发器

1) 符号：如图1.6(a)所示。图中输入信号有D—数据，和CLK—时钟，CLR—复位(清零)信号，S—置位(置1)信号，输出信号有Q和 $\overline{Q}$。

2) 操作：如表1.9所示。当CLK正跳变时，Q＝D；当CLK不是正跳变时，Q保持原状。当CLR＝1时，Q＝0；当S＝1时，Q＝1。

图1.7示出D触发器操作的时间图。图1.7(a)表明，数据线D的建立必须领先于CLK正跳变，这段时间称为建立(Setup)时间 t_S。数据线D的撤除必须滞后于CLK正跳变，这段时间称为保持(Hold)时间 t_H。如果不满足上述条件，D触发器就不能正确动作。输出信号Q的变化滞后于CLK正跳变，这段时间称为传播延迟(Propagation delay) t_P。在以后的叙述中，将忽略这些特性。如1.7(b)示出了不考虑 t_P 的操作时间图。

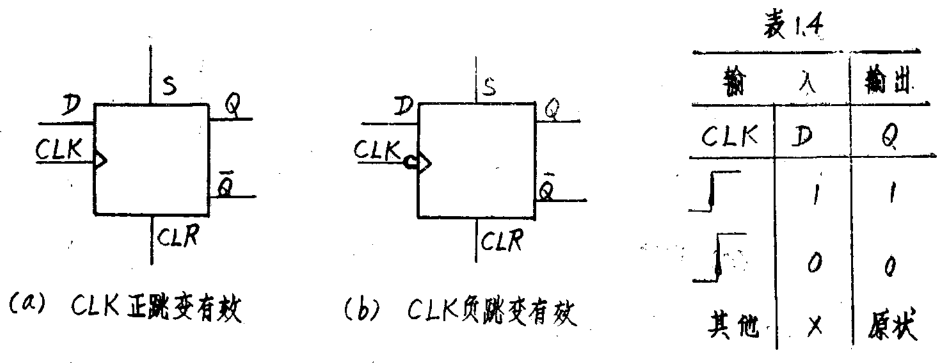


表1.4

输入		输出
CLK	D	Q
↑	1	1
↑	0	0
其他	X	原状

图 1.6 D触发器

图1.6(b)示出另一种D触发器，与前者的差别仅在于CLK端上多一个小圆圈。在数字电路中，小圆圈表示反相，因而这种D触发器的CLK信号在负跳变时为有效。同理，若CLK，S端上也加上小圆圈。则这些信号在0电平为有效。

2. JK触发器

1) 符号：如图1.8所示。J和K为控制输入端。

2）操作：表 1.5 示出了当 CLK 发生正跳变时输出与输入的关系。

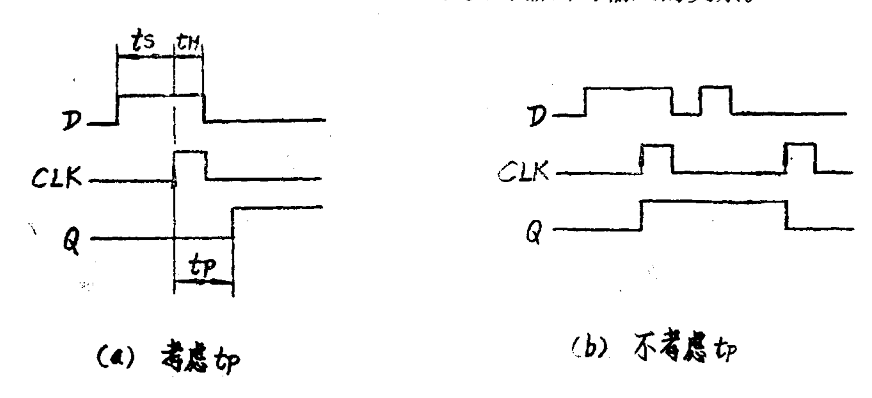


图 1.7 D 触发器的操作时间图

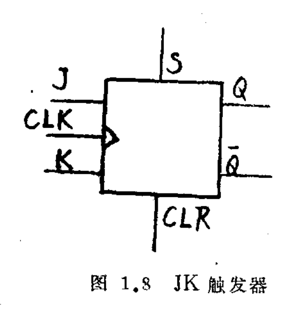


图 1.8 JK 触发器

表 1.5

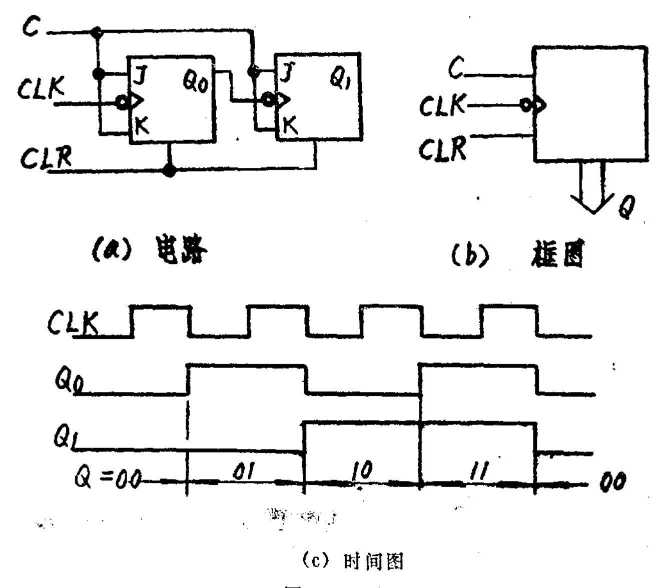
输入		输出
J	K	Q
0	0	不 变
0	1	0
1	0	1
1	1	翻 转

二、寄存器

寄存器由若干个(通常是 4，8，16 个等)触发器构成。它可以存放一个 4 位、8 位、或 16 位的字，此外还执行加 1、减 1、移位和其他的操作。

1. 计数器

1）电路图及方框图：如图 1.9(a)(b)所示。



(c) 时间图

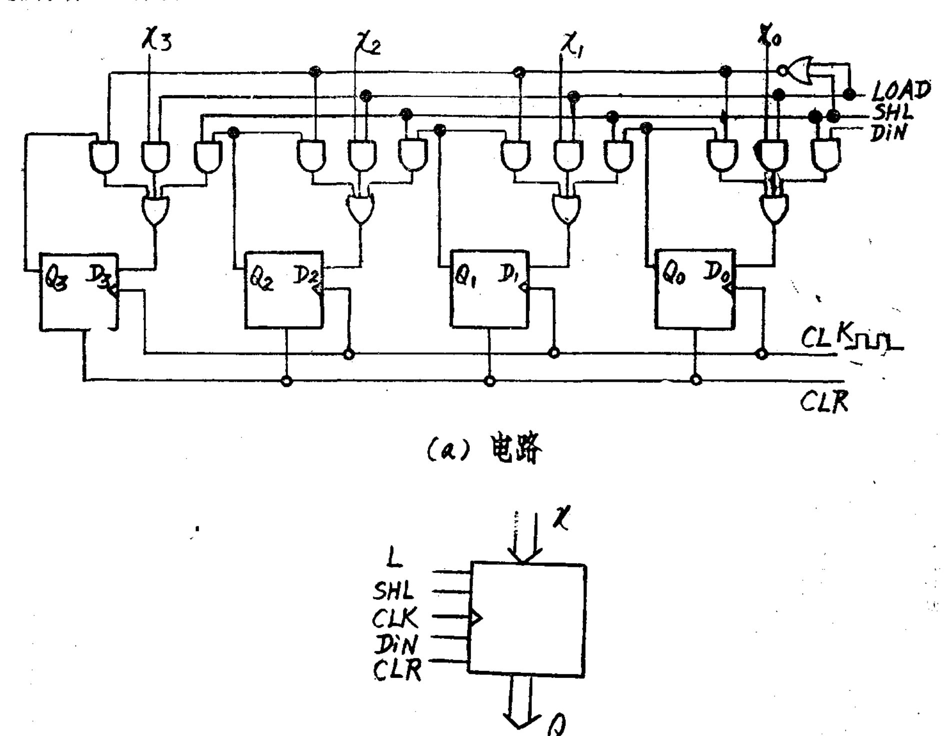
图 1.9 计数器

2）操作：当 CLR＝1 时，所有触发器被复位，即 $Q_1=0$、$Q_0=0$。当 C＝0 时，所有 J、K 端都为 0，施加时钟脉冲 CLK 并不能改变 Q 的状态，即 Q 保持原状。当 C＝1 时，所有 J、K 端都为 1，因而每一个时钟脉冲都使计数器加 1，其时间图如图 1.9(c) 所示。可知 C 是控制输入端，控制计数器是否对 CLK 进行计数。

2．缓冲与移位寄存器

1）电路图：如图 1.10 所示。

2）操作：当 CLR＝1 时，所有触发器被复位，Q＝0。当 LOAD＝1 时，在 CLK 正跳变瞬间，将 X 装入 Q，即 Q＝X。当 SHL＝1 时，在 CLK 正跳变瞬间，Q 向左移位，即 $D_{in}\to Q_0$，$Q_0\to Q_1$，$Q_1\to Q_2$，$Q_2\to Q_3$，Q_3 消失。当 LOAD＝0 和 SHL＝0 时，在 CLK 正跳变瞬间，Q 保持不变。



(a) 电路

(b) 框图

图 1.10 缓冲与移位寄存器

3．环形计数器

1）电路图：如图 1.11(a)(b)所示。

2）操作：当 CLR＝1 时，$T_3=0$，$T_2=0$，$T_1=0$，$T_0=1$，即 T＝0001。当送入时钟脉冲 CLK 时，每出现一次正跳变，T 依次所 0010→0100→1000→0001→………。如图 1.11(c)所示。

三、三态输出寄存器

由于一般的数字器件只有两个状态 1 和 0，所以每条信号线只能由一个器件来驱动，从而使信息传输线的数目大为增多。为了减少信息传输线的数目，简化控制系统，将属于

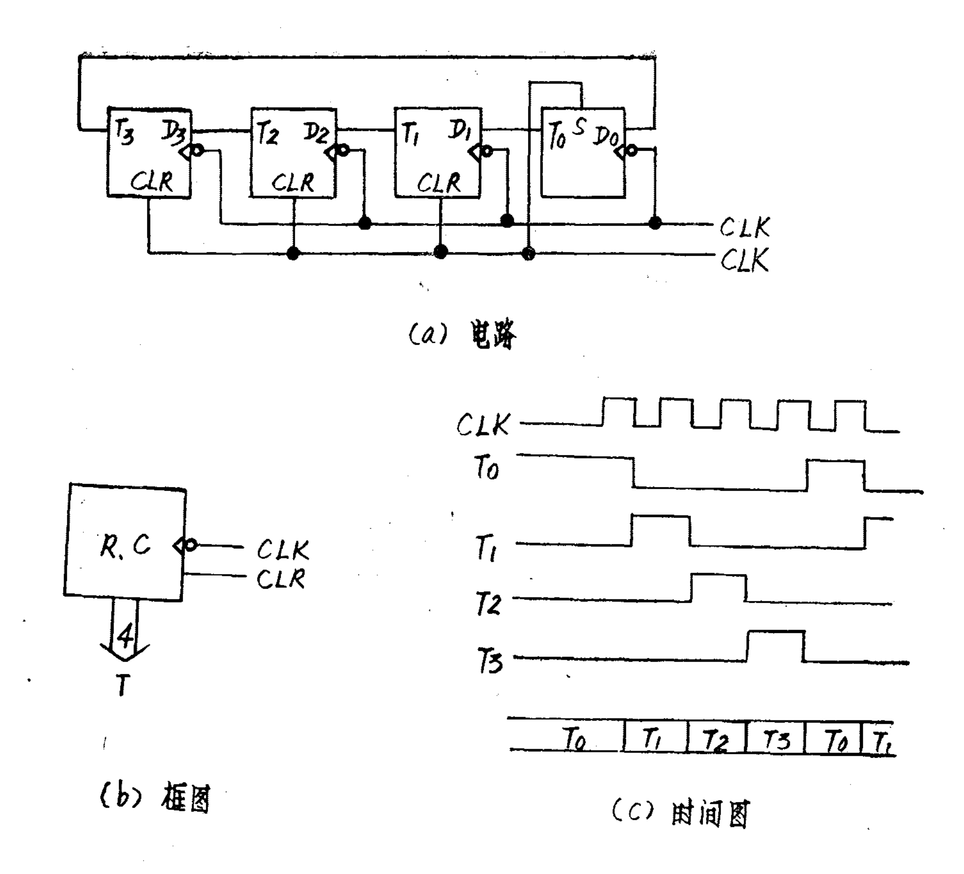


图 1.11 环形计数器

不同来源的信息或数据在一组统一的传输线上分时(Multiplexed)传送到不同的目的地，这组传输线(通常是8条或16条)称之为总线(Bus)。在某一时刻，只能有一个器件驱动总线，这个器件的输出可以是1或0。其他器件不能呈此二状态，它们必须呈第三种状态—高阻抗，即浮动状态。也就是说，好像它们的输出被开关断开，对总线状态不起作用。

为了将普通器件的二态输出更改为三态输出，通常使用图1.12所示的三态开关。当E=1时，$D_{Out} \doteq D_{in}$，此时D_{Out}线该器件来驱动。当E=0时，D_{Out}呈高阻抗状态，该器件对它不起作用，而是由其他器件驱动此线。

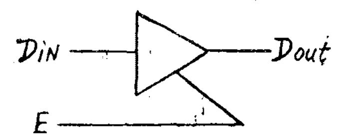
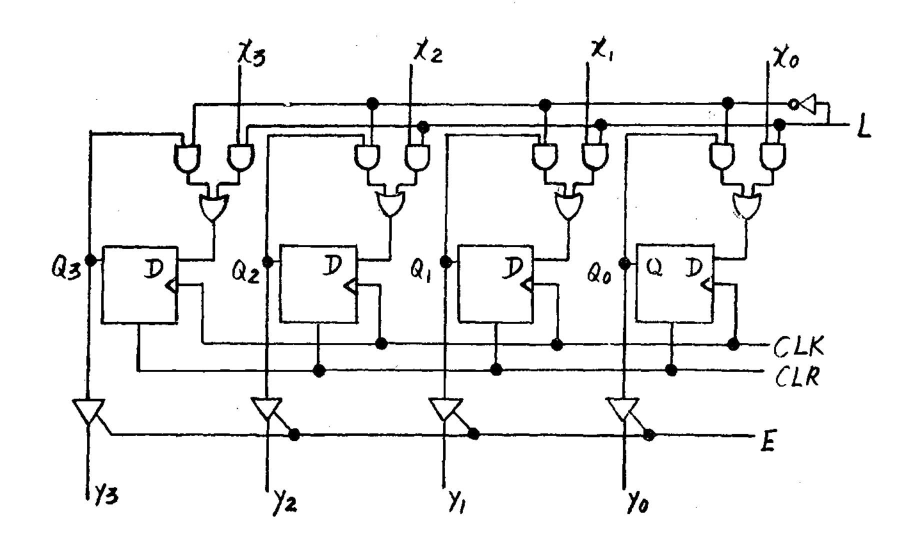


图 1.12 三态开关

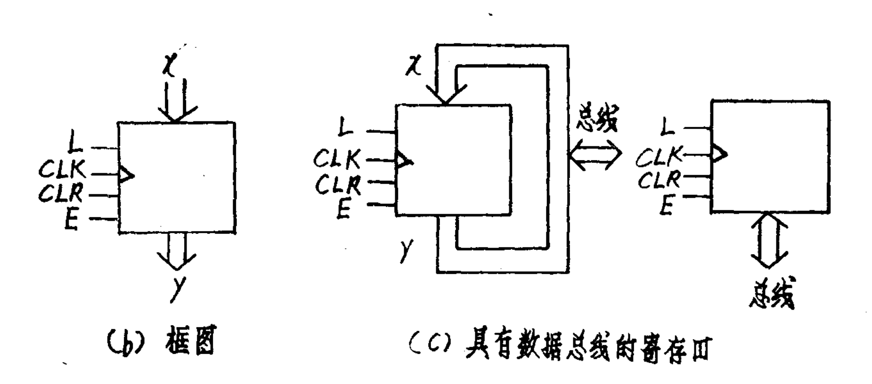
图1.13示出具有三态输出的缓冲寄存器。当E=1时，输出Y=Q；当E=0时，输出Y=高阻抗，与各个触发器的状态Q无关。对于具有三态输出的缓冲寄存器，可以将其数据输出线和数据输入线合并在一起，以便挂到总线上去，如图1.13(c)所示。

图1.14示出了四个寄存器挂在一条总线W上。如果控制信号中只有E_A、L_B等于1，其他均匀为0，则在CLK正跳变时，寄存器A的内容将装入寄存器B。因为只有E_A=1，其他器件的E信号为0，总线的状态由A来决定。换句话说，A驱动总线。因为只有L_B=1，总线上内容只装入B。通俗地说，E是“放出”控制信号，L是“装入”信号。

如果只有E_B=1，L_C=1，L_D=1，其他控制信号均为0，则在CLK正跳变时，寄存



(a) 电路



(b) 框图　　(C) 具有数据总线的寄存器

图 1.13　具有三态输出的缓冲寄存器

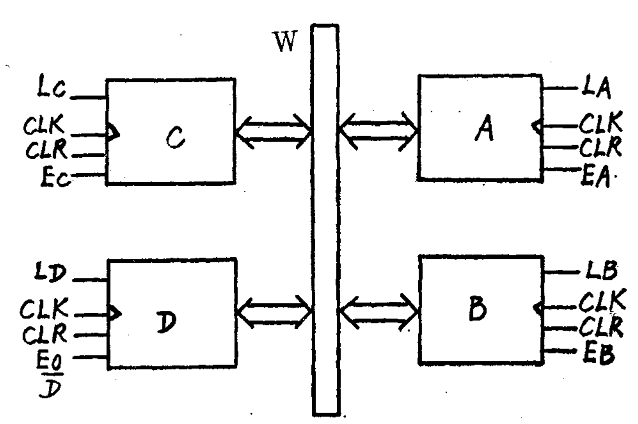


图 1.14　挂到总线上的寄存器组

器 B 的内容将装入寄存器 C 和 D。

对于某一时刻，只能有一个器件的 E 信号有效，否则将有多个器件“争夺”总线而发生误操作。

1.5 存储器(Memory)

存储器是由若干个寄存器组成，每个寄存器存放一位二进制字数(在微型计算机中，通常是8位称作一个字节)。存储器用来存放程序和数据。有多种介质可以用来制作存储器，如半导体、磁芯、磁盘、磁带等等。微型计算机的内存储器(简称内存)主要使用半导体存储器，本节内容仅限于这种半导体存储器。

存储器可以分为下列两大类：

1．只读存储器—ROM　它用来存放程序、表和常数。它的内容只能被微处理器读出，而不能被微处理器改变，也不会因断开电源而消失。ROM通常又分为下列三种：

1）ROM　它的内容由芯片制造厂使用掩模编程而被永久性地固定下来。由于其工作可靠，在大量生产时价值低廉，在产品已被定型而不量生产时，常常使用ROM。

2）PROM(可编程序只读存储器)　它内容由用户利用PROM写入器(或称编程器)而被固定下来，一旦被编定，就不能再被改变，只允许对未曾编程的位进行编程。在小批量生产时，常使用PROM。

3）EPROM(可擦除的可编程序只读存储器)　它的内容被用户编定后，可以用紫外线擦掉而再次被编程。虽然EPROM的可靠性不如前两者，但是由于它的灵活性，常被使用于产品的研制阶段。

近年来，又出现一些其他类型的ROM，如EAROM，EEPROM，此处不作介绍。

2．读写存储器—RAM　它的内容可以由微处理器写入和读出。RAM可分为两种：

1）静态RAM　每位信息存放在一个触发器中，只要电源有电，其信息能一直保持。

2）动态RAM　它利用门基片电容上的电荷来存放信息。这种电荷将在几毫秒内耗散，因而每隔两毫秒需对动RAM的内容刷新一次。

动态RAM的优点是器件密度高，价格低，待机功耗低。它的缺点是必须有刷新电路，每个动态RAM芯片的字长仅为一位，因而8位微型计算机(即使内存容量很小)要求最少使用八个芯片。

一、只读存储器 ROM

只读存储器的原理图如图1.15(a)所示。每一行可以看作一个存储单元，用来存放一个二进制字。图中共有四行，即这一存储器有四个单元，可以存放四个二进制字。图中共有四列，表示每个字的字长为4位。

如果图中的地址线 $A_1=0$、$A_0=1$、$E=1$，则第二条线 R_1 为高电平，从而通过二极管而使 D_0 为高电平，D_1、D_2、D_3 相应的列中没有插入二极管，故为低电平，即

$$D=D_3\ D_2\ D_1\ D_0=0001$$

其他线 R_0，R_2，R_3 都为低电平，故对输出D没有影响。同理，A_1，A_0 呈不同值时，选中不同的水平线呈高电平，从而使数据线上呈现不同单元的内容，有二极管的位，相应于逻辑1；反之，没有二极管的位，相应于逻辑0。

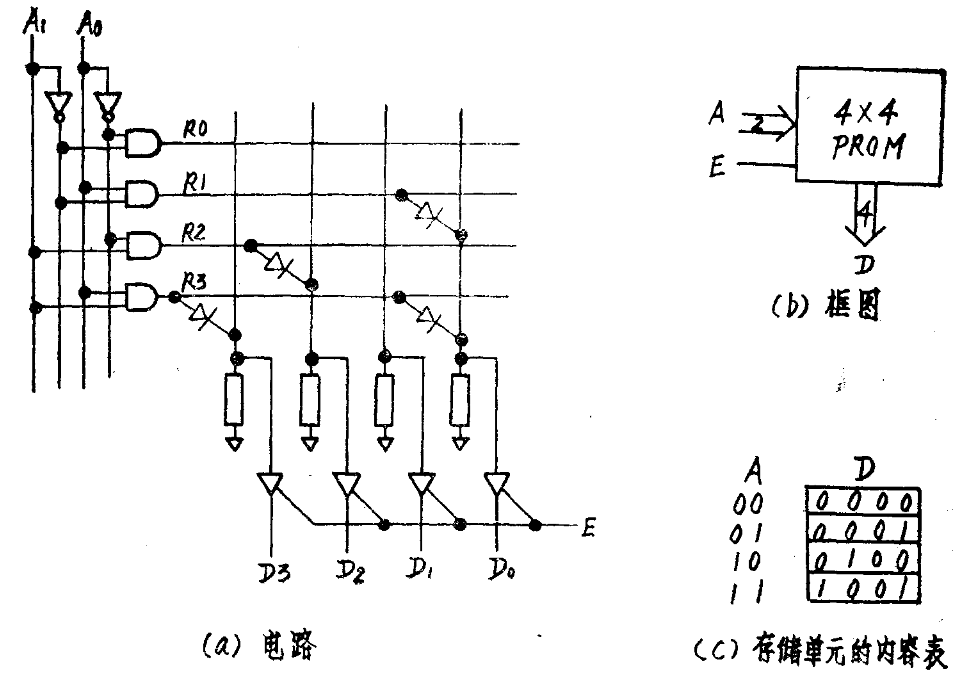


图 1.15 只读存储器的原理图

如果图中矩阵中二极管的有无可由用户来决定，而且可以再次被改变，则可以认为这是一种 EPROM 的模型。

图 1.5(b)示出它的方框图。数据输出线 D 的位数表示字长，通常是 8 位；地址输入线 A 的位数 N 决定存储字数，字数 $=2^N$。

图 1.15(c)用表格来表示 ROM。在该图中，ROM 的内容 $R=(A1A0)^2$。可见，这样的 ROM 可以用作平方的表格。推而广之，ROM 还可以存放各种函数(包括逻辑运算函数)以及其他的常数等。

图 1.16 示出了目前常用的 2716 型 EPROM(2K×8)的引脚图。在 28 根引脚中，有 11 根地址线，8 根数据输出线，以及 V_{CC} 和地线。这些引脚易于理解，不再进一步解释。现介绍其余三根引脚的作用，如表 1.6 所示。当 $V_{PP}=+5V$ 时，为读数方式；当 $V_{PP}=+25V$ 时，为编程(即写入—Programming)的方式。

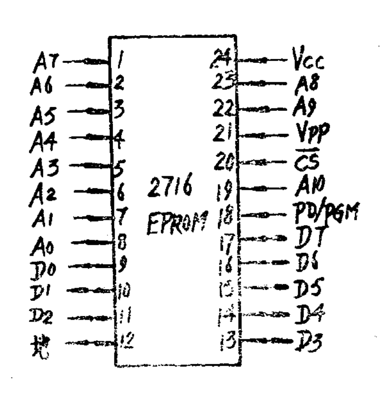


图 1.16 2716 型 EPROM(2K×8)的引脚图

表 1.6 工作方式选择

引脚 方式	PD/PGM (18)	$\overline{CS}$ (20)	V_{PP} (21)	数据线状态
读	0	0	+5V	D_{OUT}
未选中	×	1	+5V	高阻抗
待机	1	×	+5V	高阻抗
编程	⎍	1	+25V	D_{IN}
校验编程内容	0	0	+25V	D_{OUT}
禁止编程	0	1	+25V	高阻抗

下面扼要介绍这几种工作方式：

1．读——此时可将其中某一个单元的内容读至数据线上。

2．未选中——因为 $\overline{CS}=1$，数据输出线呈提阻抗，即该芯片不起任何作用。

3．待机(PowerDown)——当 PD/PGM＝1 时，芯片处于待机方式。这种方式与不选中相似，唯一的差别在于：待机方式的功耗仅为最大运动功耗的四分之一。

4.编程——若要对 EPROM 某个单元进行写入，则应对 PD/PGM 引脚输入一个正脉冲，脉冲宽度为 50 毫秒左右，对2K个单元的写入编程总共需要 100 秒左右。由于施加的脉冲电平与 TTL 兼容，不需要施加高电压脉冲，因而不必使用专门装置，而用单板机这样的简单系统即能编程。

5．校验编程内容——在编程完毕后，将其中的内容读出并进行比较，以决定编程的内容有否出错。此方式与读方式相似。

6．禁止编程——此时禁止将数据线上内容写入 EPROM。

二、读写存储器 RAM

图 1.17 示出了 21 14型静态 RAM(1K×4)的引脚图。在 18 根引脚中，有 10 根地址线，4 根数据(I/O)线，2 根为 V_{CC} 和地线。这些引脚的作用易于理解，不再进一步解释。表 1.7 说明了 $\overline{CS}$ 和 $\overline{WE}$ 的作用。

表 1.7 RAM 的操作

$\overline{CS}$*	$\overline{WE}$	操作	数据线状态
1	×	保持原状	高阻抗
0	1	读	读出单元的内容
0	0	写	简要写入的内容

* 或称为 ME 信号

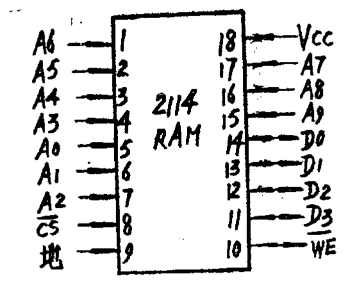


图 1.17 2114 型疾态 RAM(IK×4)的引脚图

第二章 模型计算机

在学习微型计算机和计算机的过程中，往往会有难于入门的感觉，其原因在于“太烦”。为此本章介绍一下尽可简单的模型机，在第六节还对此机作必要的扩充，以使初学者能在不到10个学时内，了解一个计算机的全貌，同时也掌握一些现代微型计算机中的重要概念，为进一步应用微型计算机技术创造条件。

2.1 模型计算机的结构

模型计算机的结构如图 2.1 所示。它是由算术逻辑运算部件 ALU、寄存器（存储器可看作多个寄存器）挂到总线上，并配以必要的控制电路而构成。这条总线既传送数据，也传送地址。如果寄存器的输出未接至总线，则为二态的；否则就是三态的。

图 2.1 模型计算机的结构

下面介绍各个部件的作用。

1. 可编程序只读存储器 PROM PROM 中存有 16 个 8 位字，即其容量为 16×8 位。它的内容和地址都可用 16 进制数来表示，如图 2.2 所示。在上面的单元中存放指令（R0—R4），下面的单元中存放数据（R9—RB）。

当 E_R 为高电平时，16 个 PROM 存储单元中有一个被读到总线上，这个单元的地址由存储器地址寄存器 MAR 来决定。

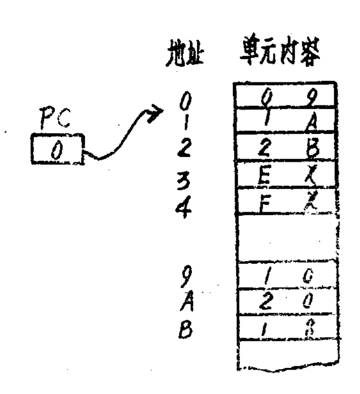


图 2.2 PROM 的内容表

2．存储器地址寄存器 MAR 这是一个 4 位奇存器，存放被访问存储单元的地址。其内容可以来自程序计算器 PC，也可以来自指令奇存器 IR 中的低 4 位(指令中的地址码)。它的二态输出送往 PROM。

3．指令寄存器 IR 这是一个 8 位寄存器。计算机在执行程序中，将指令从 PROM 中取出，放入 IR。8 位的指令码由两部分组成：高 4 位是操作码，表示该指令所应执行的操作，此信号送向 CON 单元，但不受 E_1 信号的控制；低 4 位是地址码，表示该指令执行的操作中所需的数据在哪个存储单元，当 E_1L_M 有效时，此信息将送向 MAR。

4．累加器 A 这也是一个 8 位奇存器，用来存放运算的中间结果。它有两个输出：一是送往算术逻辑单元 ALU 的二态输出，其不受 E_A 信号的控制；另一是送往总线的三态输出。

5．B 寄存器 它也是一个 8 位奇存器，接受总线上送来的加数或减数，并有一个送往 ALU 的二态输出，以供 ALU 进行运算。

6．算术逻辑运算单元 ALU 这是一个加法器/减法器(见 1.4 节)，S_U 电平的低或高决定它进行加或减。

$$当 S_U=0 时，ALU=A+B$$

$$当 S_U=1 时，ALU=A-B$$

ALU 是一种异步(无时钟的)电路，A 和 B 对它的输入都是二态的，它随时对累加器 A 和 B 奇存器进行运算。当 $E_U=1$ 时，它的内容方出现在总线上。

7．输出寄存器 O 这是一个 8 位奇存器。当计算机运算结束时，运算结果存放在累加器 A 中，需要将它传送到输出奇存器 O，以便送往外部。

8．二进制显示器 D 它是连接到输出奇存器 O 的 8 个发光二极管 (LED)，用来显示输出奇存器 O 的内容。

9．程序计数器 PC 这是一个 4 位计数器。它被用来指示当前该执行的指令的地址，如图 2.2 所示。计算机在复位后，它的内容为 0000。每当一条指令从存储器中被取出后，PC 内容自动增 1。因而，在每个取(指令)周期的起始时，PC 中保存当前指令的地址。如图 2.2 所示，PC=0000，表明该执行 0000 单元中的指令 09。

10．控制单位 CON 通过有节奏地送出 12 个控制信号和时钟信号，它指挥计算机有节奏地执行指令所要求的操作。详细介绍见 2.4 节。

上述十个部件可进一步划分为下列几个部分。

1．运算器——包括 ALU、A、B。它的任务是执行算术的和逻辑的运算。

2．控制器——包括 PC、IR、CON。它按一定的顺序从存储器取出程序的一条指令，加以“解释”，发出操作命令。此外它还要安排一定的操作步骤，使计算机中各部分在得到操作命令后，按一定的节拍，有条不紊地操作。控制器是计算机工作自动化的保证。

上述两部分又可统称为 CPU(中央处理机)。笼统地讲，微处理器就是 CPU。

3．存储器——包括 PROM、RAM。它用来存放指令和数据。与运算器和控制器直接联系的存储器称为内存。

上述三部分是计算机的主要组成部分。

4．输出设备——包括 O 和 D。用来显示所得到的计算结果。

2.2 指令系统和程序设计简介

计算机与一般电子设备不同，它可以利用上节介绍的硬件，针对不同的问题进行相应的程序设计。然后将此程序的指令逐条送入存储器，并启动计算机运行。

用户在编制程序时，必须熟悉计算机的指令系统，即计算机能执行的基本操作。

一、指令系统

模型机的指令系统如表2.1所示，它共有五条指令。指令码由8位二进制数组成。

1. LDA×(0000××××)

这是一条加载累加器A的指令。它将存储器中某单元的内容装入累加器A。指令码的高4位0000表示操作的性质，称为操作码；指令码的低4位××××表示要加载A的数的地址，即操作数的地址，称为地址码。表2.1中所示的LDA9(00001001)指令表示将存储器中9单元的内容R9装入累加器A。

用二进制数(如00001001)表示指令能够被计算机“理解”，所以称为“机器语言”。在书写过程中，使用二进制数既烦，又易出错，故常使用16进制数(如09_H)。但是，利用机器语言编写的程序很不直观，如09_H这一数据，很难看出它是指令还是一个数。即使知道它是指令的话，也很难知道它是什么指令。因此人们就想办法，利用一些意义明确的符号来表示指令。例如09_H指令，可用“LDA9”来表示。由LDA可知，这是一条加载累加器A指令(Loa DAccumulator)。这种符号(LDA)含义明显，便于记忆，故称之为助记符(MNEmonic)。在此基础上发展出一种具有一定语法规则的符号语言，称为汇编语言。遗憾的是，对于同样的指令，不同的计算机有不同的助记符。

表2.1 模型计算机的指令系统

汇编语言		机器语言				操作及说明
		二进制		十六进制		
助记符	操作数	操作码	地址码	操作码	地址码	
LDA	9	0000	1001	0	9	A←R9
ADD	A	1001	1010	1	A	A←A+RA
SUB	B	0010	1011	2	B	A←A-RB
OUT		1110	××××	E	×	O←A
HLT		1111	××××	F	×	停止

2. ADD 0001××××

它将累加器A的内容加上存储中某指定单元的内容，其结果仍放在累加器A。同样，指令码的低4位表示操作数的地址。

3. SUB 0010××××

它将累加器A的内容减去存储器中某指定单元的内容，其结果仍放在累加器A中，同样，指令码的低4位表示操作数的地址。

以上三条指令统称为访问存储器指令，因为这些指令的操作都与存储器中某指定单元

内的操作数有关。

4. OUT　1110××××

它将累加器A的内容送往输出寄存器O。指令码的低4位可以任意值，指令的操作与此值无关，即不涉及存储器中存放的操作数。

5. HLT　1111××××

它停止计算机的运行。同样，指令码的低4位可为任意值，指令的操作与此值无关，即不涉及存储器中存放的操作数。

二、程序设计简介

程序是一个人为编定的指令序列，用来解决用户提出的各个问题。编写程序的过程称为程序设计。下面我们举例说明，如何编写程序，使计算机来计算16D+32D-24D=？

首先，将这些需要运算的数据安排在存储器的数据区，通常安排在高地址单元中，如表2.2所示。本例中这三个操作数存放在R9，RA，RB。

由于计算往往从0单元开始并顺序执行指令，因此第一条指令安排在0单元中。

0单元：LDA9将R9单元的内容16D装入累加器A。执行完后，A=16D。

1单元：ADDA将RAM单元的内容32D加到累加器A，结果存入累加器A。A=16D+32D=48D。

2单元：SUBB将累加器A中的内容减去RB：结果存入A。A=48D-24D=24D。

3单元：OUT累加器A中的内容送入输出奇存器O。O=A=24D。

4单元：HLT使计算机停止运行。在本模型中，将禁止时钟脉冲的发出。

表2.2　程序设计举例

地　址	机器代码	汇编语言	操作结果
0	0 9	KDA　9	A=10H
1	1 A	ADD　A	A=30H
2	2 B	SUB　B	A=18H
3	E×	OUT	O=18H
4	F×	HLT	
9	1 0		
A	2 0		
B	1 8		

操作结果如表2.2所示。

在存储器中，存放指令区必须与存放数据区分开，以免顺序执行指令时，误将数据当作指令来执行，从而产生荒谬的结果。

大多数微型计算机都能将汇编语言翻译成机器语言，这种过程称为汇编过程(Assembly)，使用的工具称为汇编程序或汇编器(Assembler)。

机器语言和汇编语言都与机器本身有关，是一种面向机器的低级语言。使用这种语言编制程序不够方便。且不能在机器间交流程序。因而人们又不断设计出面向问题或面

向过程的高级语言(本书中不作介绍)。

2.3 指令的执行过程

一条指令的操作不是一下子就能完成的,而是如同人们做操那样,在好几个节拍内完成的。每一个节拍完成一个微操作,这些微操作的集合即是指令所要求的操作。本机中每条指令需要六个节拍来完成,正如人们做操需要八个节拍一样。如表 2.3 所示,前三拍为取周期,即将指令从存储器中取出。T_0 拍将 PC 的内容送 MAR,因为当前要执行的指令的地址放在 PC 中。T_1 拍将 MAR 指定的存储器单元的内容(指令码)取出,放在 IR 中。T_2 拍使 PC 内容增 1,为执行下一条指令作准备。取周期的微操作与要执行的指令的种类无关,五条指令在 T_0—T_2 节拍内执行的微操作相同。后三拍为执行周期,其微操作因指令不同而各异,有些指令只需一个或两个节拍即可完成其要求的操作。例如 ADDA 指令,T_3 拍将 IR 中低 4 位(即地址码)送 MAR。T_4 拍将存储器中的加数(其地址由 MAR 规定)送 B 寄存器,此时 ALU 中的内容为累加器A 和 B 寄存器相加的和。T_5 拍将 ALU 中的和送累加器 A。其他指令的具体过程见表 2.3,不再赘述。

不同的指令在不同的节拍内要执行不同的微操作。为此,在此节拍内除了需要有 CLK 正跳变外,尚需要有相应的控制信号,如表 2.3 所示。

2.4 控制单元 CON 的设计

计算机能有条不紊地工作,全靠 CON 的指挥。

计算机对 CON 的要求如下:

1. 产生清零脉冲 CLR;
2. 产生时钟脉冲 CLK;
3. 循环产生六个不同的节拍 T_0—T_5;
4. 在不同指令的不同节拍内产生所需的控制信号;
5. 启动计算机,使之从 0_H 单元开始执行程序。

按照上述要求设计的控制单元 CON 的电路如图 2.3 所示。其中各个部件的功能如下:

1. 复位按钮 RESET　按下此按钮,即向计算机其他部件发 CLR 信号。

2. 启动按钮 ST 和时钟电路　当按下 ST 按钮时,Q1 信号由低变高,从而使 Q2 信号由低变高。当 Q2 为高电平时,CLK 不停地发出时钟脉冲,当 Q2 在低电平时,CLK 保持低电平,没有出现时钟脉冲。

3. 环形计数机 RC　其结构与图 1.11 相似,所不同的是它由六个触发器构成。当 CLR 信号有效时,T_0 呈高电平,以后每当出现 CLK 负跳变时,轮流使 T_1, T_2, …… 为高电平。

4. 指令译码器 ID　如图 2.4 所示,其输入信号来自指令寄存器 IR 的高 4 位,当 PC＝0001 时,输出信号 ADD 应为高电平。其余输出信号均为低电平。余此类推。

表 2.3 指令的执行过程

指令 \ 节拍	指令周期					
	取周期			执行周期		
	T0	T1	T2	T3	T4	T5
LDA 9 0000 1001	MAR←PC L_M, E_P	IR←R(MAR) L_I, E_E	PC←PC+1 C_P	MAR←IR L_M, E_I	A←R(MAR) L_A, E_B	—
ADD A 0001 1010	MAR←PC L_M, E_P	IR←R(MAR) L_I, E_R	PC←PC+1 C_P	MAR←IR L_M, E_I	B←R(MAR) L_B, E_R	A←A+B L_A, E_U
SUB B 0010 1011	MAR←PC L_M, E_P	IR←R(MAR) L_I, E_R	PC←PC+1 C_P	MAR←IR L_M, E_I	B←R(MAR) L_B, E_R	A←A—B L_A, E_U, S_U
OUT 1110 ××××	MAR←PC L_M, E_P	IR←R(MAR) L_I, E_R	PC←PC+1 C_P	O←A L_O, E_A	—	—
HLT 1111 ××××	MAR←PC L_M, E_P	IR←R(MAR) L_I, E_R	PC←PC+1 C_P	— HLT	—	—

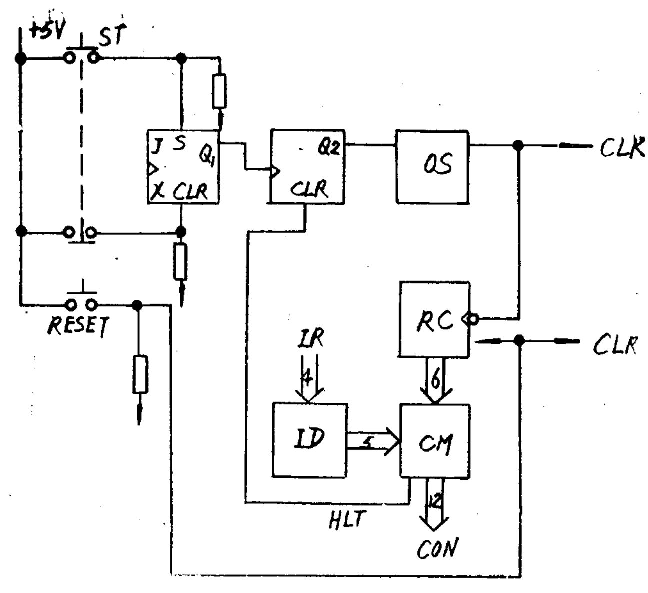


图 2.3 控制单元CON的电路

5. 控制矩阵CM 这个部件用来在不同指令的不同节拍内产生不同的控制信号(又称不同的控制字CON)，如表2.3所示。或者说，它使输入信号 I_i 和 T_i 与输出信号CON之间成一定的逻辑函数关系 $CON = f(I_i, T_i)$。表2.3即为这种函数的格值表。

图2.5为控制矩阵的电路图。以 L_M 信号为例，在下列四种情况下均需出现 L_M 信号。

1) T0节拍：不管什么指令，只要T0线为高电平，则 L_M^1 为高电平，L_M 为高电平。

2) LDA指令的T3节拍：此时 $LDA = 1$，$T3 = 1$，由于 $L_M^2 = 1$，$L_M = 1$。

3) ADD指令的T3节拍：同理，亦可得 $L_M^3 = 1$，$L_M = 1$。

4) SUB指令的T3节拍，此时亦可得 $L_M^4 = 1$，$L_M = 1$。

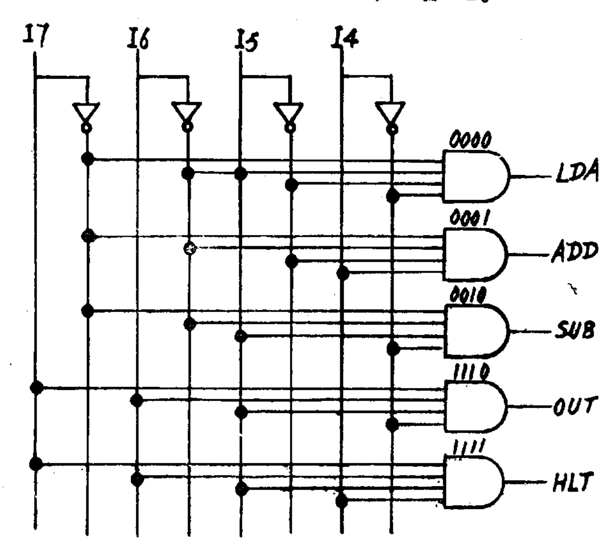


图 2.4 指令译码器ID的电路

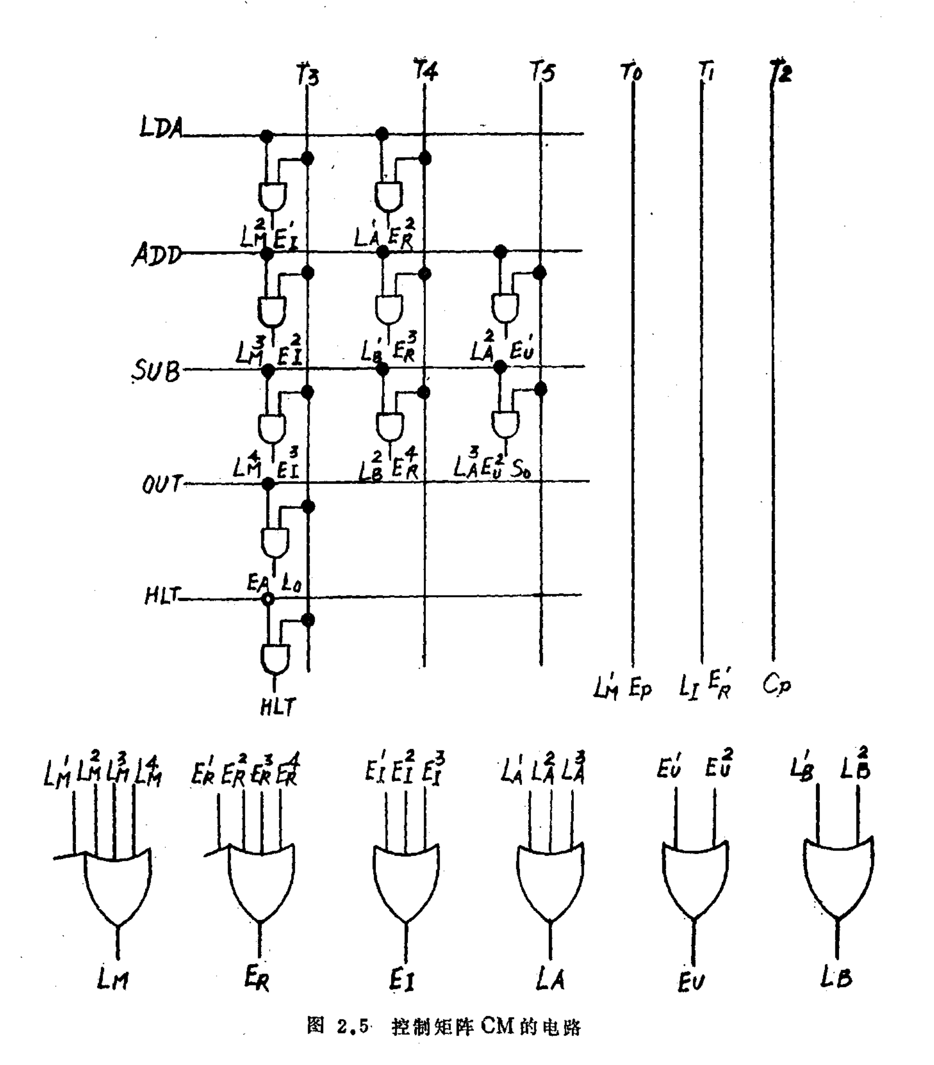


图 2.5 控制矩阵CM的电路

2.5 模型计算机的操作

PROM 的内容如表 2.2 所示。模型计算机操作的时间如图 2.6 所示。

首先压下 RESET 按钮，发出 CLR 信号，使环形计数器的 T0 成高电平，并使 PC 和 IR 为 0000。接着压下 ST 启动按钮，CLK 不停地发脉冲，环形计数器的 T0—T5 反复地轮流出现高电平，即轮流出现不同的节拍。值得注意的是，环形计数器电平的转换发生在 CLK 的负跳变瞬间。下面叙述此后发生的过程：

1．T0 期间：L_M，E_P 为高电平，在 CLK跳变瞬间，MAR←PC，所以 MAR＝0000。

2．T1 期间：L_I，E_R 为高电平，在 CLK 正跳变瞬间，IR←R(MAR)，所以 IR＝09H。

3．T2 期间：C_P 为高电平，在 CLK 正跳变瞬间，PC←PC＋1，所以PC＝0001。

4．T3 期间：L_M，E_I 为高电平(因为指令寄存器 IR 的高 4 位为 0000，指令译码器的 LDA 线成高电平 从而使控制矩阵的 L_M，E_I 为高电平)，在 CLK 正跳变瞬间，MAR←IR，所以 MAR＝1001。

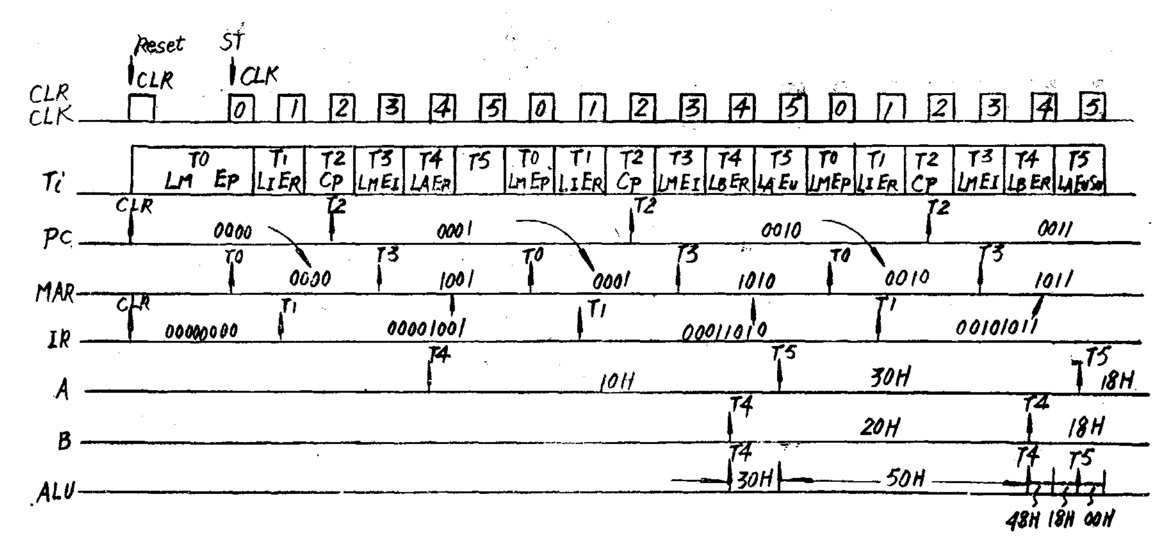


图 2.6 模型计算机操作的时间图

5. T5 期间：L_A，E_R 为高电平，在 CLR 正跳变瞬间，A→R(MAR)，所以 A = R9 = 0001，0000 = 10H。

6. T5 期间：没有发生控制信号，不进行任何操作。

此后，又不断出现 T0—T5 以执行后继单元的指令，其过程相似，不再赘述。

顺便指出，某些时刻 ALU 中的内容毫无意义，只要它不影响计算机的正确操作即可。

2.6 扩充模型机

以上叙述的模型计算机十分简单，非常便于了解计算机工作的全貌及其主要特点。但是它有相当的局限性，有一些重要概念无法介绍。为此，设计了下述的扩充模型机。

一、结构与指令系统

如图 2.7 所示，其结构与前述的模型计算机相似，仅有如上差别：

1. 存储器为 RAM 而不是 PROM，容量为 256×12，为了便于写入，添加一个存储器数据寄存器 MDR。

2. 用 SC(栈)代替 PC，SC 内有四个寄存器 SC(0)，SC(1)，SC(2)，SC(3) 用一个 2 位的可逆计数器(可以进行加 1 和减1)SP 来选择其中一个 SC(SP) 进行上述 PC 的工作。SP 称为栈指示器。如图 2.8 所示。

例如，若 SI = 00，则选中 SC(0)进行 PC 的工作。若 SP = 10，则选中 SC(2) 进行 PC 的工作，其余类推。

PU 信号用来使 SP 在 CLK 正跳变时加 1，PD 信号用来使 SP 在 CLK 正跳变时减 1。

* 在教学中本节可省略不讲

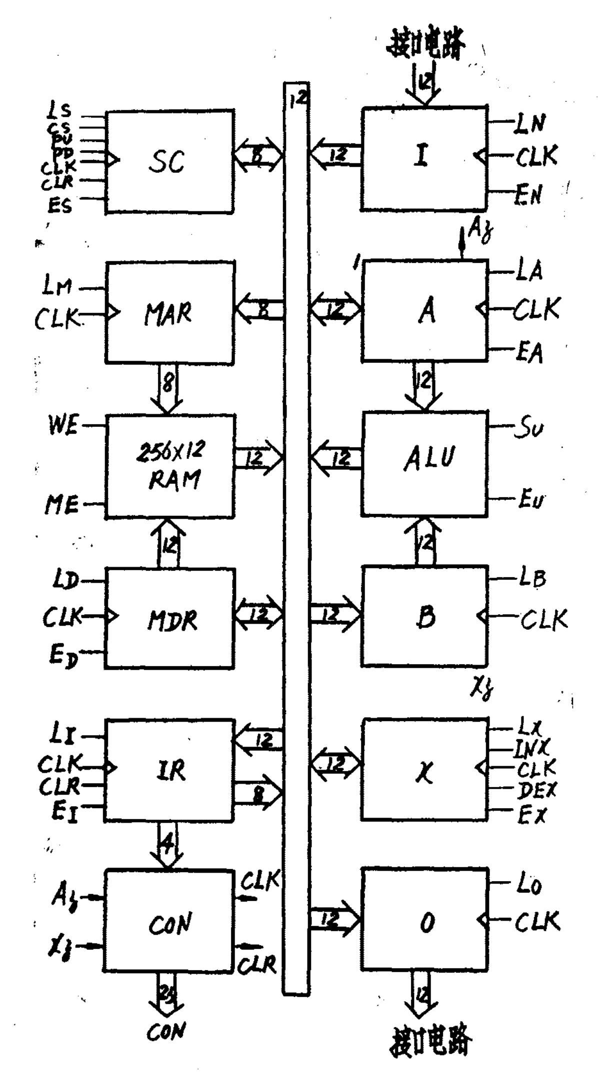


图 2.7　图扩充模型机的结构

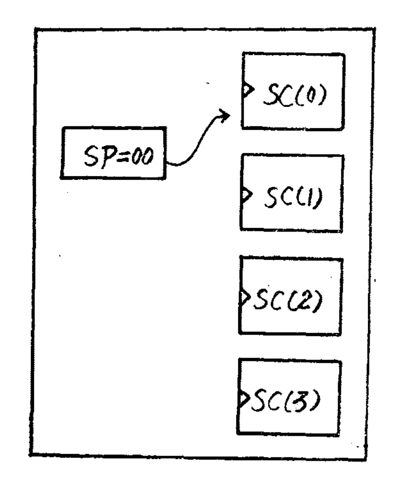


图 2.8　栈 SC 的结构示意图

3．增加了变址寄存器X。若信号NX=1，则X在CLX正跳变时加1，若信号DEX=1，则X在CLK正跳变时减1。它的作用在下面予以介绍。

4．增加了输入寄存器I。外部设备通过I将信息送入RAM。

5．总线为12位。

扩充模型机的指充系统见表2.4。指令中的地址码使用一个任意数字，以便阐明指令的操作。

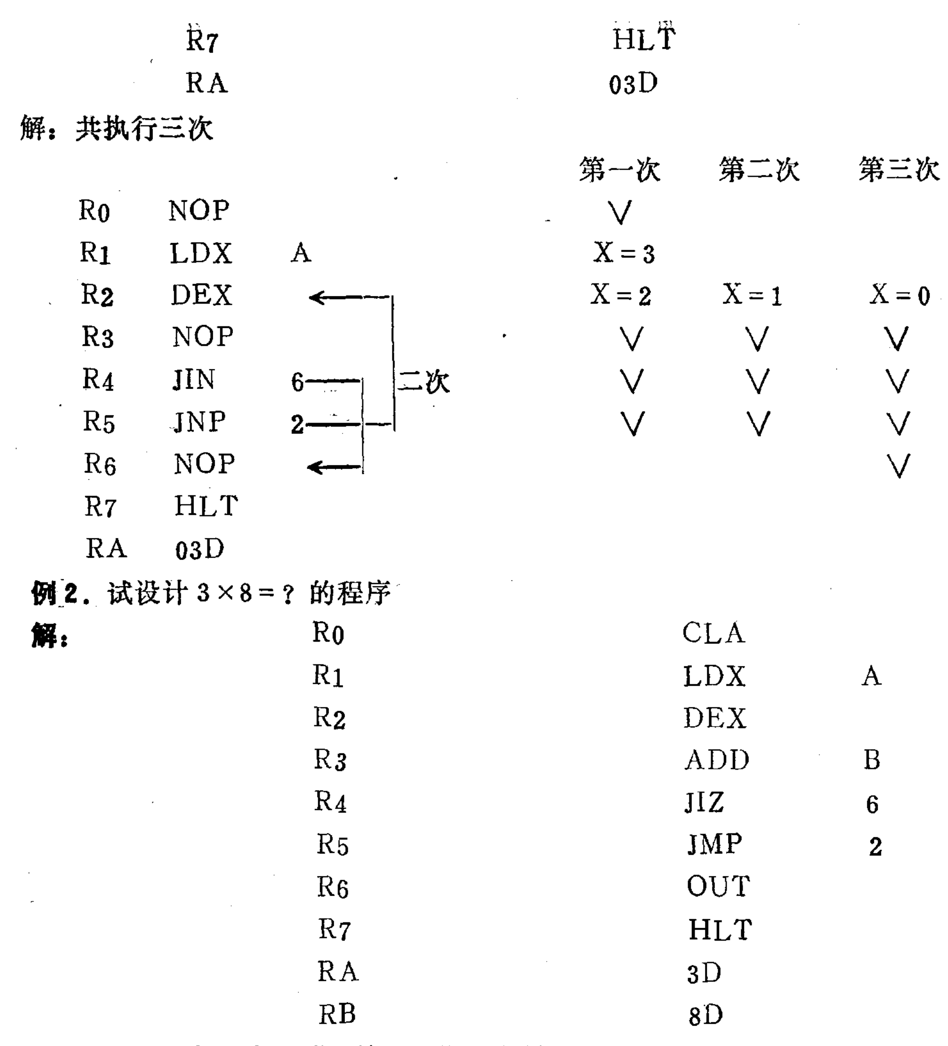
表 2.4 扩充模型的指令系统

汇编语言		机器代码 二进制			十六进制	操作及说明
助记符	操作数	操作码		地址码		
LDA	32	0000	0011	0010	032	A←R32
ADD	33	0001	0011	0011	133	A←A+R33
SUB	34	0010	0011	0100	234	A←A−R34
STA	3B	0011	0011	1011	33B	R3B−A
NOP		0100	××××	××××	4××	不操作
LDX	43	0101	0100	0011	543	X←R43
DEX		0110	××××	××××	6××	X←X−1
INX		0111	××××	××××	7××	X←X+1
JMP	56	1000	0101	0110	856	PC←56
CLA		1001	××××	××××	9××	A←000
JIZ	5E	1010	0101	1110	A5E	若X=0，PC←5E
JMS	63	1011	0110	0011	B53	SP←SP+1 SC(SP)←63
BRB		1100	××××	××××	C××	SP−SP−1
IMP		1101	××××	××××	D××	A←I
OUT		1110	××××	××××	E××	O←A
HLT		1111	××××	××××	F××	停

二、程序设计举例

例1．试问上列程序中R2—R4的指令段执行多少次？

R0	NOP	
R1	LDX	A
R2	DEX	
R3	NOP	
R4	JIZ	6
R5	JMP	2
R6	NOP	



R7	HLT
RA	03D

解：共执行三次

				第一次	第二次	第三次
R0	NOP			V		
R1	LDX	A		X=3		
R2	DEX	←	二次	X=2	X=1	X=0
R3	NOP			V	V	V
R4	JIN	6		V	V	V
R5	JNP	2		V	V	V
R6	NOP	←				V
R7	HLT					
RA	03D					

例 2．试设计 $3\times8=?$ 的程序

解：

R0	CLA	
R1	LDX	A
R2	DEX	
R3	ADD	B
R4	JIZ	6
R5	JMP	2
R6	OUT	
R7	HLT	
RA	3D	
RB	8D	

改变例 1 中 R0，R3，R6 单元中指令即得。

例 3．试设计一程序以求 $4^2+6^2+8^2=?$

题中有三次要求一个数的平方，为此编写一个求平方的子程序（从 RA2 单元开始）。每当求平方时调用此子程序即可，而不必重复三次编写“求平方”的程序，从而可简化程序设计，并节省存储单元的空间。

子程序中最后一条指令 BRB 并不要求给出地址码（即使能给出，也是一个变化值，在编写程序时不易表示），那末子程序是如何回到主程序中刚才离开的地方？将在下面予以介绍。

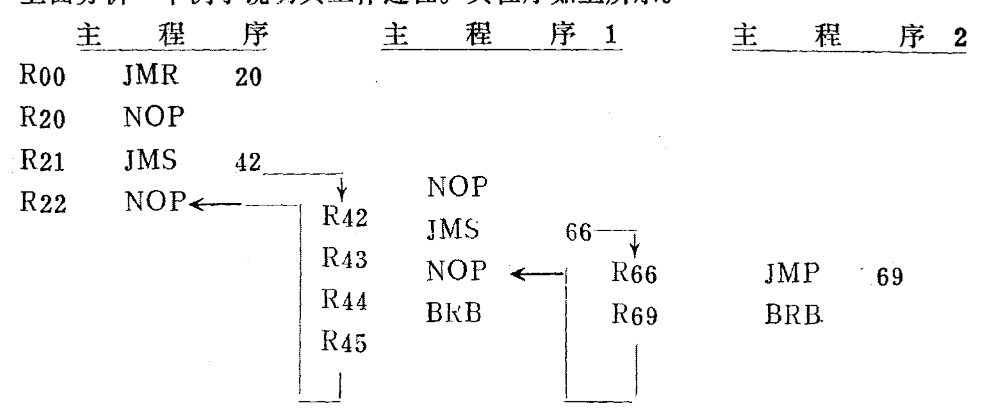
三、栈的结构 SC

SC 中有四个寄存器 SC(0)，SC(1)，SC(2)，CS(2)，SC(3)。它们中的任何一个都可象PC那样的工作。每个节拍内只能由 SP 来选中一个进行 PC 的工作。例如 SP=00，则由 SC(0)进行 PC 的工作，如果 SP=10，则由 SC(2)进行 PC 的工作。SP 是一个二位的可逆计数器。当 PU=1 时，在 CLK 正跳变时 SP 增 1；反之，当PD=1 时，在 CLK 正跳变时 SP 减 1。在复位时 SP=0。

:

主程序			子程序		
R00	LDA	C6	RA2	STA	CA
R01	JMS	A2	RA3	LDX	CA
R02	STA	C9	RA4	CLA	
R03	LDA	C7	RA5	DEX	
R04	JMS	A2	RA6	ADD	CA
R05	ADD	C9	RA7	JIZ	A9
R06	STA	C9	RA8	JMP	A5
R07	LDA	C8	RA9	BRB	
R08	JMS	A2			
R09	ADD	C9	RC6	4D	
R0A	OUT		RC7	6D	
R0B	HLT		RC8	8D	
			RC9		
			RCA		

上面分析一个例子说明其工作过程。其程序如上所示。



这个程序的执行过程如上表所示：

当子程序执行 R21 单元中“转子”(转子程序)JMS 指令时，主程序上一条要执行的指令的地址保存在 SC(0)中，此后暂停使用 SC(0)。在执行子程序 1 时，使用 SC(1) 作为程序计数器；当执行 R45 单元中 BRB 指令时，只要恢复使用 SC(0)作为程序计数器，即可找到返回的地址。子程序1 转子程序 2，以及返回的过程与此相同。SC 中有四个计数器，因此子程序可嵌套三级。这种结构是执行“转子”指令时，从硬件下保护程序计数器的内

节拍 指令	T0 MAR←SC	T1 IR←R(MAR)	T2 SC←SC+1	T3	T4	T5
R00 JMP 20	MAR=SC(0) =00	IR=R00 =820	SC(0)=01	SC(0)=20		
R20 NOP	MAR=SC(0) =20	IR=R20 =4××	SC(0)=21			
R21 JMS 42	MAR=SC(0) =21	IR=R21 =B42	SC(0)=22	SP←SP+1 SP=01	SC(1)=42	
R42 NOP	MAR=SC(1) =42	IR=R42 =4××	SC(1)=43			
R43 JMS 66	MAR=SC(1) =43	IR=R43 =B66	SC(1)=44	SP←SP+1 SP=10	SC(2)=66	
R66 JMP 69	MAR=SC(2) =66	IR=R69 =869	SC(2)=67	SC(2)=69		
R69 BRB	MAR=SC(2) =69	IR=R69 =C××	SC(2)=6A	SP=SP-1 SP=01		
R44 NOP	MAR=SC(1) =44	IR=R44 =4××	SC(1)=45			
R45 BRB	MAR=SC(1) =45	IR=R45 =C××	SC(1)=46	SP←SP-1 SP=00		
R42 NOP	MAR=SC(0) =22	IR=R22 =4××	SC(0)=23			

容，Intel 4040 即是采用这种结构。现代大多数微型机都在 RAM 中开辟一个栈区，将程序计数器的内容保护在栈中，从而可以使 SC 中只有一个计数器 PC，如同模型机那样。这种方法的优点是：子程序可以嵌套很多级，并且还可以将 CPU 中一些工作寄存器的内容保护进栈，以免在执行子程序时破坏主程序运算中所得到的中间结果。栈区的工作过程将在下章叙述。

2.7 模型计算机系统的简化

图 2.1 所示的模型计算机可以看作由三个部件组成。每个部件制做在一个芯片内，也就是三个芯片即可构成一个计算机系统，如图 2.9 所示。这三个芯片为：

1. CPU—包含 A，ALU，B，PC，IR，CON

2. 存储器—PROM，MAR

3. 输入输出—I，O

这种模型计算机传送数据和地址分时地(Multiplexed)共用一条总线。但只有极少数微型机(如 Intel 4004/4040 和 F8)使用这种工作方式，大多数微型机使用两条总线分别传送数据和地址。后者的优点是可以缩短微型机的指令周期，即提高其运算速度；缺点是芯片下使用更多的引脚，从而增大器体的体型尺寸。简而言之。微型机系统就是若干个芯片(CPU、RAM，I/O)挂到三条(数据、地址、控制)总线下而构成。如图 2.10 所示。

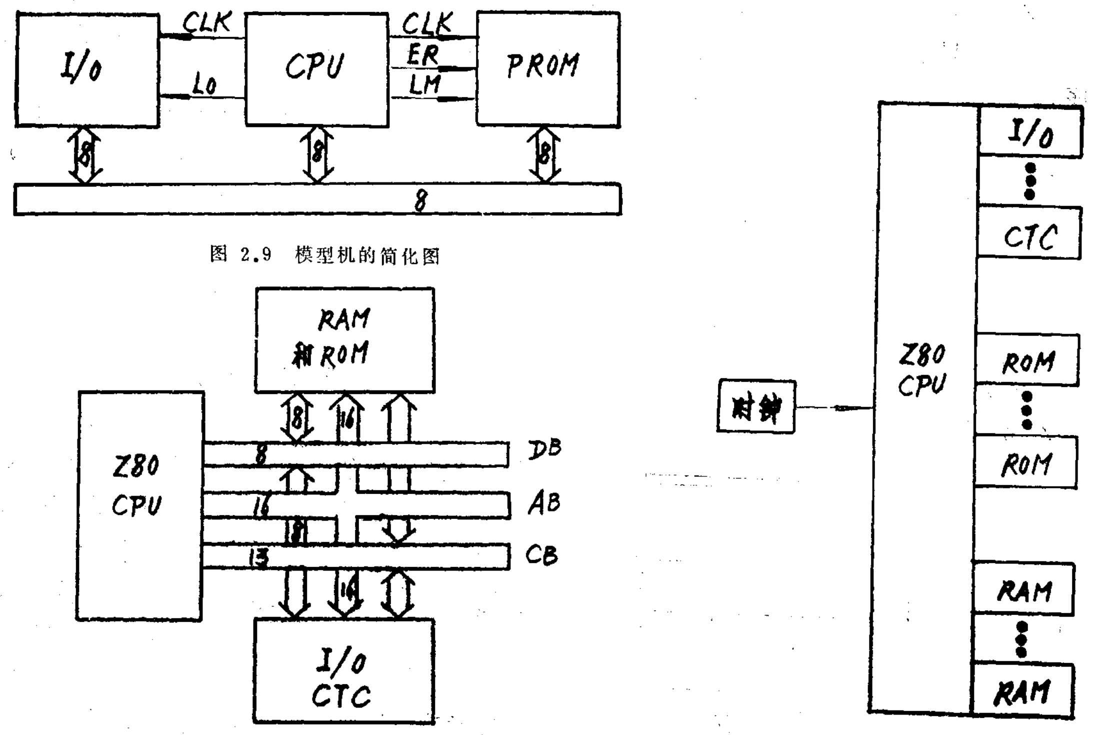


图 2.9 模型机的简化图

图 2.10 一般的微型机系统—以 Z80 为例

第三章　Z80-CPU 的结构

CPU 是中央处理单元(Central Processing Unit)的简称。它是微型计算机的中枢，担负处理数据和控制整个微型计算机系统的使命。

Z80—CPU 是指由美国 Zilog 公司原型设计生产的微型计算机系统中的超大规模集成电路 CPU 芯片。根据厂家所制定的指令系统，在使用这种芯片时用户可以直接编写程序。

为了研究 Z80—CPU 的结构，我们需要建立它的程序模型，从编写程序的角度来观察组成 CPU 的各部分。为了掌握 Z80 程序的编写方法，我们将指令系统视为 CPU 的传递函数，逐条理解其中每个指令的执行在 CPU 硬件结构中所引起的变化。

3.1 CPU　在微型计算机系统中的作用

因为 CPU 的结构和指令系统是根据对于 CPU 功能的要求来设计的，所以在具体介绍Z 80—CPU 之前，有必要先来考察它在整个微型计算机系统中的作用。

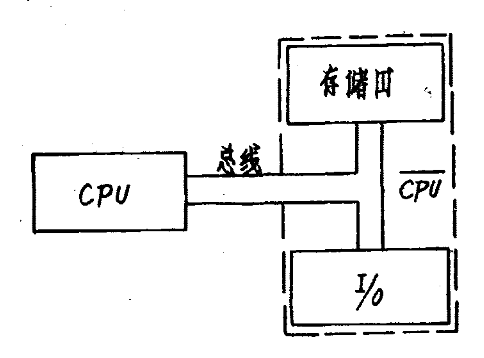


图 3.1　微型计算机系统的简化框图

如图 3.1 所示的微型计算机系统，除 CPU 外，还有存储器和输入/输出器件 (I/Odevice)，都由 $\overline{CPU}$ 表示。CPU 的功能可以概括为对数据的**读**、**写**(CPU与 $\overline{CPU}$ 之间)、**运算**(CPU内部)，**传送**(CPU 内部以及与 $\overline{CPU}$ 之间)，以及 CPU 对它本身及对 $\overline{CPU}$ 的**控制**。所有上述四种功能都是通过对于数据字或控制字的传送和处理来实现的。这些字的每 8 位(bit)字长叫做一字节(byte)，每 4 位字叫做半字节(nibble)。

在 CPU 内部的数据传送主要是通过软件设计(程序编写)来操纵的，在 CPU 和 $\overline{CPU}$ 之间的数据传送主要是通过硬件设计(电路接口)来实现的。

3.2　Z80—CPU 的结构

Z80—CPU 是在 8080A 微处理器的基础上改进设计的。主要的改进是：

1．减少芯片数目。为了有效使用 8080ACPU，还需配用 8224 时钟发生器和 8228 系统控制器。Z80—CPU 将三种芯片的功能综合在一起，减少了芯片数目，为用户提供便利。

2．减少电源种类。8080A 需要三种电源(＋5V，－5V 和 ＋12V)，Z80—CPU 只需要一种＋5V 电源。

3．其他功能方面主要是指令系统功能的加强，包括更强有力的中断功能，变址寻址功能，数据块传送功能，位操作功能，动态存储器刷新功能等。

Z80—CPU 的结构如图 3.2 所示，主要由寄存器、运算器、指令译码及定时和控制几个部分组成。

一、寄存器

在 CPU 的整个功能中，寄存器扮演最重要的角色。寄存器实质上是一个小的存储器，它由一定数目的二态存储单元所组成，本身具有一定的地址以便加以识别。

在 Z80—CPU 中，包含有 208 位可供用户使用的寄存器，全部用静态 RAM 实现。它们分为专用和通用两类，其程序模型如图 3.3 所示。

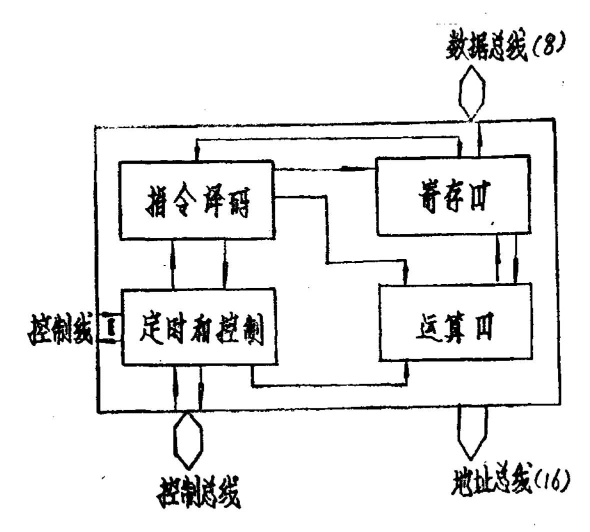


图 3.2 Z80—CPU 的简化框图

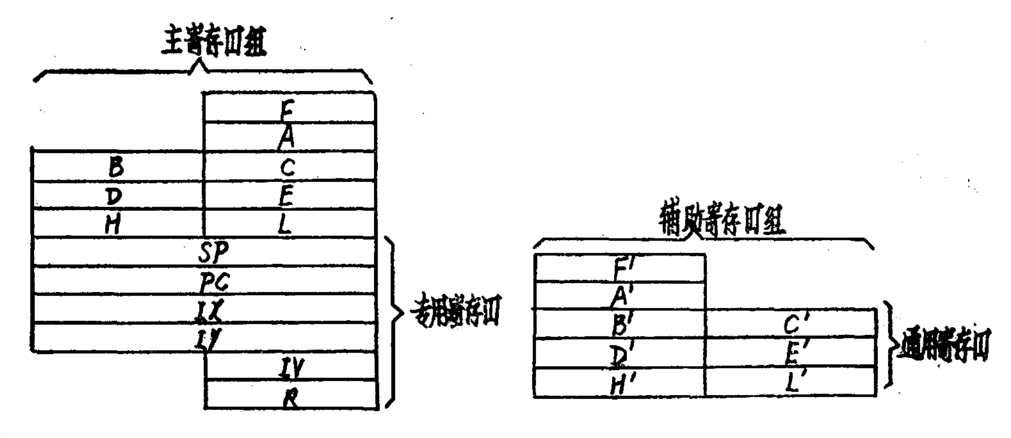


图 3.3 Z80—CPU 的寄存器

Z80—CPU 中的专用寄存器包括：

1．程序计数器(Program Counter，简称 PC)。其内容是将要执行的下一指令字节所在的存储单元的地址。一个指令周期开始时，CPU 将 PC 的内容放在地址总线上，使 CPU 从存储器中取出指令的第一个字节。CPU 继而使 PC 内容增1，使 PC 指向相继的指令

字节。

2. 栈指示器(Stack Pointer，简称 SP)。用于贮存某一指向特定存储单元地址的寄存器叫做指示器。指示器的内容，即所贮存的地址叫做指针。

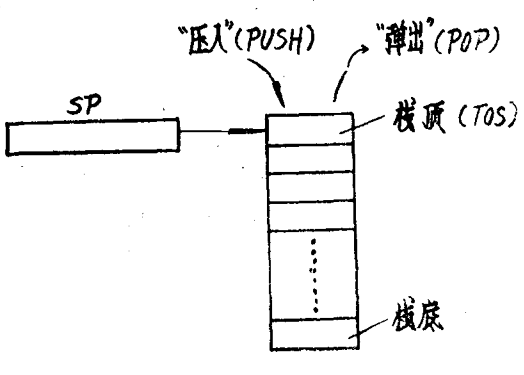


图 3.4 栈与栈操作

栈是一种暂存数据(或地址)的存储区，好象一个货栈。组成栈的一些存储单元，象一沓碟子，有规律地排列着。最先进栈的数据子构成栈底，栈的另一端为栈顶(Top of Stack，简称 TOS)。对于用户来说，只有最后进栈的一个数据字，即处于栈顶的一个数据字才能被存取(见图 3.4)。

从软件的角度看，栈是一种数据结构。其中各元素的内容和数目都在动态地变化，但变化方式不是随意的，而是先进后出(First In Last Out，简称 FILO)的。所有的元素进栈或出栈都要通过栈顶。中间的元素不能跳出这个序列。因此栈也叫做下推表。

在栈操作中，栈中的各元素实际上并未移动。唯一变化发生在栈指示器 SP 中。SP 是一个专用 16 位寄存器，用于贮存栈顶地址。当一个数据字进栈时，需将 SP 增 1(或减1)，栈顶上升，数据字存放在 SP 增量后所指向的新栈顶。这种操作叫做"压入"(PUSH)。如果要从栈中取出数据，则最先取出已经处于栈顶的数据字，然后将 SP 减1(或增1)，降下栈顶，并依此类推。这种操作叫做"弹出"(POP)。

3. 变址寄存器(Index Register)。在 Z80—CPU 中，设有两个完全相同的变址寄存器 IX 和 IY。这是一种贮存 16 位地址的寄存器。其内容不仅可以在程序控制下循环增 1 或减 1，而且能与指令包含的一个操作数(叫做偏移，offset)相加，形成一个新的有效地址，指向所要访问的单元(见图 3.5)。在处理数组和表时，使用这种寄存器尤为便利。

4. 中断矢量寄存器(Interrupt Vector register，简称 IV 或 I)。使微型计算机暂停正常程序流程，接受输入信号的请求去执行一定的服务子程序(称为中断服务程序)，然后再返回原来的程序流程，这种工作方式叫做中断(见图 3.6)。

为了能够准确返回原来的程序流程，需要自动将中断前一时刻各 CPU 寄存器的内容保护起来，称为保护现场；而在执行完中断服务程序之后再恢复这一现场。此外，为了使 CPU 在响应中断后能自动转向中断服务程序，需要形成中断服务程序的入口地址，也叫

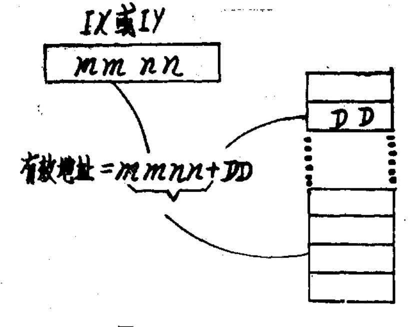


图 3.5 变址操作示意图

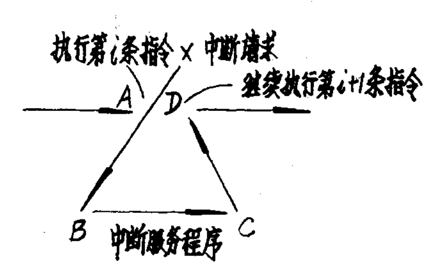


图 3.6 中断操作示意图

做中断矢量。在 Z80—CPU 中，为用户提供了形成中断矢量的三种方式，方式 0，方式 1 和方式 2。在方式 2 中，借助于 8 位的中断矢量寄存器 IV 来形成中断矢量是最完备的一种。如图 3.7 所示，当 Z80—CPU 工作在这种方式时，需要制定一个 16 位中断矢量地址表。这个表可以位于可寻址的存储器中的任何部位。其中各登记项所组成的16 位地址标识各中断服务程序中第一条可执行的指令，即入口地址。当 CPU 在方式2的条件下响应中断时，请求中断的外部设备应将一个中断矢量的低 8 位放在数据总线上去。Z80—CPU 将 IV 的内容与这个矢量结合起来，形成一个 16 位地址。用这个地址，就能够访问中断地址矢量表，进入所需要的中断服务程序。

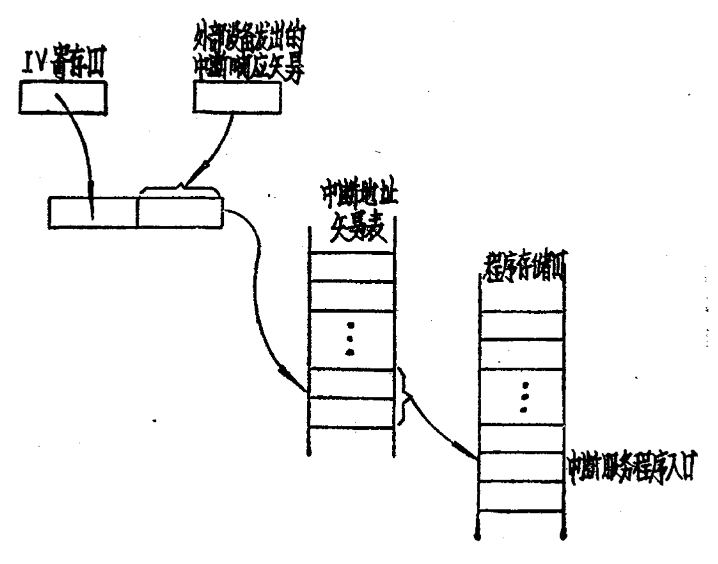


图 3.7 中断矢量的形成

5. 存储器刷新计数器(memory Refresh counter，简称R)。在 Z80 微型计算机系统中，允许采用十分廉价的动态 RAM。但是，除非大约每 2ms 进行一次刷新，否则在这种存储器中所贮存的信息就会消失。为了解决这个问题，需要能够周期性地访问动态 RAM，在每次访问中都将各存储单元内容重新写入的动态刷新电路。在 Z80—CPU 中的存储器刷新计数器就是为此而设置的。

6. 累加器(Accumulator，简称 A 或 Acc)和标志寄存器 (Flag register)。在 CPU 中累加器是使用最频繁的一种寄存器。在 CPU 完成各种运算期间，累加器作为中间结果的暂存器；在各种8位算术运算中，总有一个操作数存放在累加器中。CPU 还利用累加器来实现逻辑操作，移位及某些指令所要求的特殊操作。

标志寄存器由六个 1 位标志触发器组成(图 3.8)。它们反映 CPU 内部的某种操作状态，保存或反映运算的部分结果；有的标志还能输送补充加数 (如 CY 标志)。在 Z80—CPU 中，设有两类标志。一类主要为 CPU 形成判定操作提供测试条件，叫做测试标志；另一类主要用于 BCD 运算，叫做非测试标志。测试标志用来作为执行转移、调用或返回指令的条件。根据对某个标志位状态测试的结果是1还是 0，就能够决定是否执行相应的操作。测试标志包括：

1) 进位标志(Carry，简称 C 或 CY)。如果最后一次运算从最高位(MSB)产生进位，则将其存入此标志。在做减法时，最高位产生借位时此标志也能置1。此外，各种移位指

令也能影响此标志。

2）零标志(Zero，简称 Z)。如果最后一次运算结果为零，则此标志置1；否则置 0。

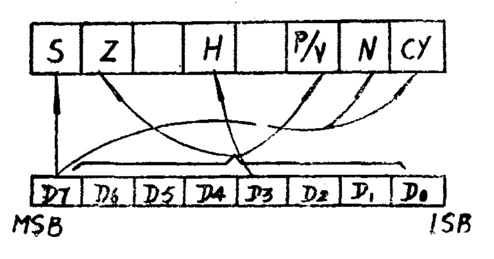


图 3.8 Z80—CPU 的操志寄存器

3）符号标志(Sign，简称S)。在有正负号的运算中，用数据字的最高位表示正负号。如果最高一次运算的结果为负数，S标志置 1；否则置 0。

4）奇偶/溢出标志(Parity/oVerflow，简称 P/V 或 P/O)。这个标志标识运算结果的奇偶性，如果结果中有偶数个 1，则P/V标志置1否则置0。这个标志还标识是否发生二的补码溢出。如果发生溢出，则此标志置 1；否则置 0。

对于一个数据字节，如用最高位表示正负号则有 7 个有效位，能表示 -128—+127 范围内的数。如果运算的结果超出了这个数值范围，就会发生溢出，以两个数M，N相加为例，设其最高位分别为M_7，N_7；结果为R，其最高位为R_7，则有下列真值表：

M7	$\bar{N}7$	R7	P/V
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

若两个数同号，和的符号却不同，说明发生了溢出

若M，N具有不同符号，不会发生溢出

若两数同号，和的符号相同，说明没有发生溢出

判别减法是否发生溢出，要看参加运算数的符号相反的情况，其真值表如下：

M7	N7	R7	P/V
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

若M，N具有不同符号，差的符号与被减数相反，说明已发生溢出

故可列出产生溢出条件为：

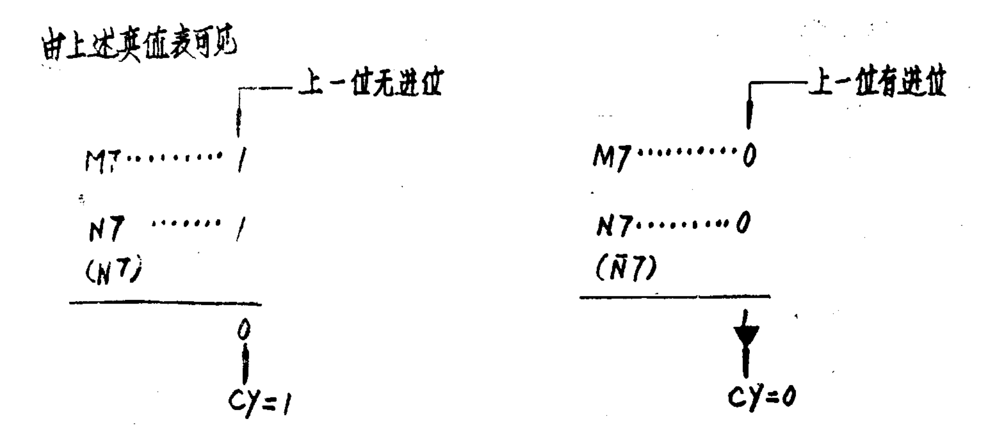
上一位有进位送入最高(符号)位，从最高位无进位输出；

上一位无进位送入最高(符号)位，从最高位有进位输出。

将此条件写成表达式：

$$P/V = d_{bcy} + CY$$

式中 d_{bcy} 表示从和的第 6 位向最高(第 7)位的进位。



Z80—CPU 的非测试标志有两个：

1）半进位标志 H)alfcarry 简称 H)。当低位半字节向高位半字节产生 BCD 进位或借位时，此标志置 1。这个标志用于校正 BCD 加法或减法的结果。

2）减标志(Negate，简称 N，此标志也叫做加/减标志，Add/Subtract flag)。在 BCD 运算时，十进制调整指令 DAA 利用此标志来区别加法和减法。(前面进行的一次)操作是加法，N置 0；(前面进行的一次操作)是减法，N 置 1。

累加器和标志位都可以由一些特定的指令来直接访问或受到影响。对于某几种涉及寄存器对指令(如 PUSH，POP 指令)，累加器和标志寄存器被视为一个寄存器对，叫做程序状态字(Program Status Word)或处理器状态字(Processor Status Word，简称 PSW)。在 Z80—CPU 中，有两组独立的累加器 A，A′ 和标志寄存器 F，F′；组成 PSW 和 PSW′。

Z80—CPU 的通用寄存器也叫做工作寄存器(Working reg isters)。它们具有多种功能，并能在程序的直接操纵下工作。利用通用寄存器，可以暂存参加运算的操作数和中间结果，避免中间结果在存储器和累加器之间来回传送。从这个意义上讲，它们也叫做次级累加器。另外，通用寄存器也可以存放数据地址，作为数据计数器使用。这时就需组成寄存器对，作为专门指向数据存储区的指示器。

在 Z80—CPU 中有两组独立的 8 位通用寄存器，主寄存器组(main register set)B，C，D，E，H，L 可以进行 8 位操作，也可以组成对进行某些16位操作。辅助寄存器组(alternative register set)B′，C′，D′，E′，H′，L′ 的排列与主寄存器组一一对应。如前所述，当 Z80—CPU 响应中断后需要保护现场；待中断服务程序结束后需要恢复现场。如果只有一级中断申请的话，就不必再设栈区。只用一条指令就能将主寄存器内容送入辅助寄存器，然后在需要时再交换回来。这样就能大大简化中断操作。因此，辅助寄存器也可以看成是设置在内部的栈区。

二、运算器

运算器也叫做算术和逻辑单元 CPU(Arithmetic & Logic Unit，简称 ALU)。其主要功能是完成各种算术和逻辑运算。Z80—CPU 除具有基本加、减、比较、增 1、减 1、“与”、“或”、“异”等运算功能外，还具有丰富的移位功能以及对单个位的处理(对于各寄

存器的任一位的置、复位、测试)功能。

三、指令译码及定时和控制

这是整个 CPU 的控制中枢。它将程序存储器送来的指令翻译成控制信号，以产生所需要的动作；它使 CPU 能够找到为指令所要求的贮存在存储器和寄存器中的地址或数据；它向 ALU 提供控制输入，使之执行指令所规定的运算；它监视外部控制信号，并产生适当的响应；它为存储器和 I/O 器件提供状态，控制和定时信号等。总之，它的功能是在正确的时刻，将信息准确地送往某个目的地。它动作的依据是该时刻执行的指令，它所产生的作用是发往 CPU 和 $\overline{\text{CPU}}$ 各部分的控制和定时信号(见图 3.9)。

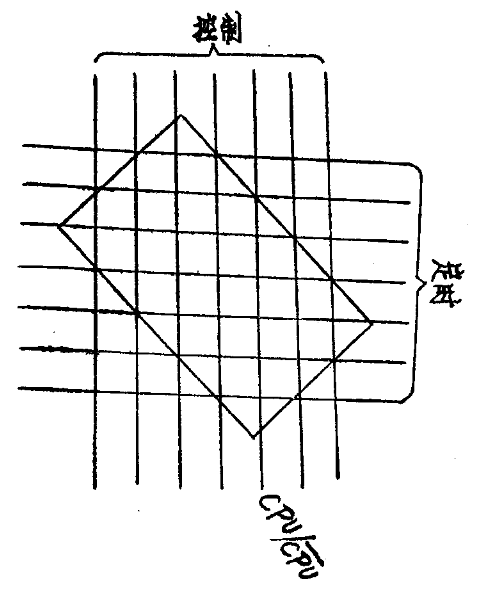


图 3.9 控制器的功能

Z80—CPU 的动作是周期性的，需要精确定时。基本定时脉冲由外部振荡器产生，接至 CPU 的 φ 输入端。如图 3.10 所示，在两个定时脉冲上升沿之间的持续时间是一个时钟周期(Clock Period)，它等于机器处于一种状态的持续时间，因此也叫做 T 状态(Tstate)或 T 周期(Tcycle)。CPU 实现某种规定的基本操作所需时间叫做机器周期(Machine cycle)或 M 周期，一般为 3 或 4 个 T 状态。有些指令在执行自动插入或 $\overline{\text{CPU}}$ 通过 $\overline{\text{WAIT}}$ 信号/插入等待(WATI)状态，这时就可能有5至 6 个 T 状态。一条指令从取出到执行完毕的持续时间叫做指令周期(Instruction cycle)。根据指令内容的不同，Z80—CPU 执行一条指令可能需要一至六个机器周期。

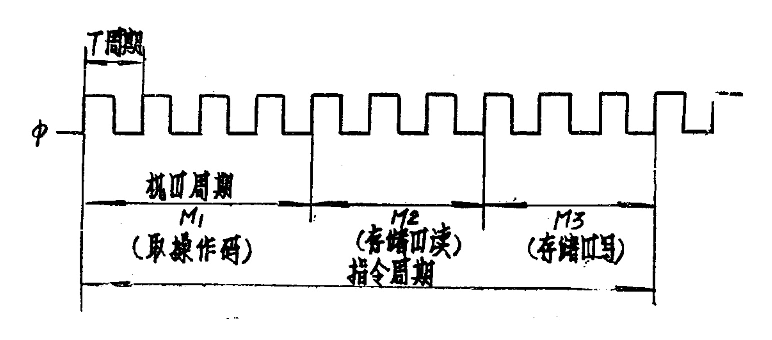


图 3.10 Z80的定时波形

四、总线

在 CPU 内部各单元之间的数据传送是通过内部总线实现的。对于用户来说，有三条总线使CPU与 $\overline{\text{CPU}}$ 相联系。

1. 数据总线(Data Bus，简称 DB)。它是 8 位双向总线。担负 $\overline{\text{CPU}}$与 CPU 的数据

传输。

2. 地址总线(Address Bus，简称 AB)。它是由 CPU 向外发出的 16 位单向总线，总共可以选择 $2^{16}=65536$ 个不同的地址。

3. 控制总线(三态线)及定时、控制线(输入或输出线)见图 3.11 的引脚图。其中，定时档控制信号引线分为三组：系统控制(6 个)，CPU 控制(5 个)和 CPU 总线控制(2 个)。所有涉及这些引线的信号都是低电平有效的。

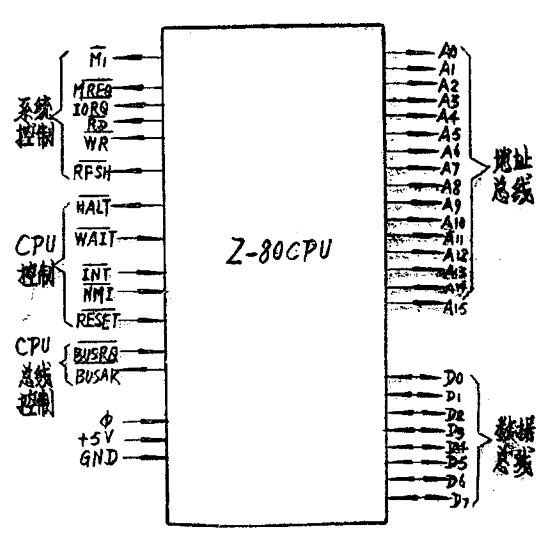


图 3.11 Z80—CPU 的引脚图

涉及系统控制方面的信号控制 CPU/$\overline{\text{CPU}}$ 之间的数据传送。其中的信号包括：

M_1(Machine cycle 1，机器周期 1)。输出。它表示在一条传令的执行期间处于取指令的机器周期。

$\overline{\text{MREQ}}$(Memory REQuest，存储器请求)。三态输出。此信号表示地址线上存在一个用于存储器读或写操作的有效地址码。

$\overline{\text{IORQ}}$(I/O REQuest，输入/输出请求)。三态输出。表示此时地址总线的低 8 位 A_0—A_7 含有一个有效的 I/O 口地址。中断响应操作在 M_1 有效期间出现。$\overline{\text{IORQ}}$ 也被用于中断响应。$\overline{M_1}$ 和 $\overline{\text{IORQ}}$ 同时存在表示中断已被接受，在数据总线上有一个中断响应矢量存在。

$\overline{\text{RD}}$(memory ReaD，存储器读)。三态输出。标志 CPU 希望从存储器或 I/O 设备中断出数据。

$\overline{\text{WR}}$(memory WRite，存储器写)。三态输出。表示出 CPU 希望向存储器或 I/O 设备写入数据。

$\overline{\text{RFSH}}$(ReFre SH，刷新)。输出。此信号表示出地址总线的低 7 位 A_0—A_6 是动态存储器的刷新地址。当前的 $\overline{\text{MREQ}}$ 信号被用于动态存储器的刷新。

涉及 CPU 控制方面的信号影响 CPU 操作的进程。其中的信号包括：

$\overline{\text{HALT}}$(HALT state，暂停状态)。输出。在执行一条 HALT 指令后，$\overline{\text{HALT}}$ 输出低电平。这时 CPU 进入暂停状态。在此期间，它连续执行 NOP 指令，以维持暂停状态，又不中断对动态存储器的刷新。

$\overline{\text{WAIT}}$(WAIT state，等待状态)。输入。等效于 8080A 的 READY 输入。如果存储器或 I/O 设备在指定时间内来不及响应 CPU 的访问请求，则它们将 $\overline{\text{WAIT}}$ 输入拉成低电平。作为对 $\overline{\text{WAIT}}$ 的响应，CPU 进入等待状态。

$\overline{\text{INT}}$(INTerrupt request 中断请求)。输入由 I/O 设备产生的中断请求信号。

$\overline{\text{NMI}}$(NonMaskable Interrupt request，不可屏蔽中断请求)。输入。负跳沿触发中断请求信号。与 $\overline{\text{INT}}$ 相象，只是比前者具有更高的优先权。不受中断允许触发器状态的影响，因此不能被禁止。

$\overline{\text{RESET}}$(RESET，复位)。标准复位控制输入。使CPU进入起始状态，使PC,IV和R寄存器清零，置中断方式为0，及禁止通过$\overline{\text{INT}}$输入的中断请求。所有的三态总线信号都被置入浮动状态。

涉及CPU总线控制方面的信号有两个。当外部设备请求占用微型计算机的数据、地址总线和三态控制总线时，外部设备将$\overline{\text{BUSRQ}}$(BUS ReQuest，总线请求)输入拉成低电平在现行机器周期结束时，CPU将使所有的三态总线进入浮动状态，并用$\overline{\text{BUSAK}}$(BUS AcKnowledge，总线响应)输出低电平来表示接受外部设备对占用总线的请求。这一对信号多用于实现直接存储器存取(DMA)。

在图3.11中，ϕ为时钟输入端，+5V是CPU要求的电源，GND为接地端。

数据、地址和控制总线都是三态的。当它们处于浮动状态时，外部设备占用对总线的控制。

第四章　Z80-CPU 指令系统

指令是 CPU 借以控制 CPU 内部各单元以及 CPU 各部分协调动作的命令，CPU 所具有的全部指令组成指令系统。因此指令系统实际上全面描述了 CPU 的功能。

为了操纵微型计算机实现某项指定任务，按照一定的规则排列的指令序列叫做程序。根据一定的算法，从指令系统中选取所需的指令，赋予一定参数后加以排序，就得到需要的程序。这个过程叫做编写程序。

对于计算机本身，用二进制代码书写的程序最易于接受。但对于用户，却宁可用符号代替指令码。在微型计算机的一般应用中，最广泛使用汇编语言指令。用这种指令编写的程序，叫做汇编语言源程序。这个程序中的各条指令经过一种专用程序——汇编程序被翻译成二进制代码的形式，叫做结果代码。整个源程序也就被翻译成为用二进制代码表示的目标程序这种翻译过程叫做汇编。

如前所述，Z80 是在 8080A 基础上设计的。在它的指令系统中，除包括了 8080A 的全部 78 条指令外，还增加了 80 条指令，总共有 158 条。

汇编语言指令用汇编语句形式书写的。为了能编好程序，首先必须了解语句语法。

指令所载有的实际信息与指令代码的格式关系很大。在学习指令系统时，每条指令的代码格式也是需要我们了解的。

每种实际微型计算机的指令系统，都是由生产厂家制定的。因此，总是有差异的。为了尽快掌握一种微型计算机的指令系统，重要的不在于单纯背诵各条指令，而在于掌握指令的分类，善于剖析典型指令的格式和操作内容。这样，才能达到举一反三，事半功倍的良好效果。

4.1　指令的语句、语法、代码格式和分类

一、语句语法

汇编语言指令是按照下列规则编定的。任何指令都具有四个独立的和不同的部分，叫做字段(field)。例如

①	②	③	④
ALTR3B:	LD	A, (DSMEM7);	GET NEW VALUE

是从某程序中取出的一条指令，其中

① 标号段(label field)：它是一个名字，用来标识 16 位的指令地址。在程序中的一条指令是否具有标号，视需要而定。也就是说，标号是任选的(Optional)。标号中打头的字符可以是字母表中的字母、a 符号、？等，但不得用阿拉伯数字。在标号的末尾须接一个冒号“：”。

② 操作码段(Operation code field)：它规定所需实现的操作。为了便于记忆，一种操作用该操作的(英文)名称中的几个字母表示的，叫做助记符。例如，助记符LD就是LoaD(传送,加载,装入)的简称；JP表示Jump(转移)操作；CP表示ComPare(比较)操作等。

③ 操作数段(Operand field)：即参与操作的数据。它与操作码一起，确定指令所要执行的操作。根据操作码段的要求，操作数段可以空白、只有一项，或者具有用逗号分开的两项。

在Z80指令中，有四种操作数：

寄存器(register)

存器对(register pair)

立即数据(immediate data)

16位存储器地址(16 bit memory address)

这些操作数可以有以下几种表示方式：

十六进制数据(Hexadecimal data)

十六进数据(Decimal data)

八进制数据(ctal data)

二进制数据(Binary data)

当前程序计数器$(current program counter)

ASCII常数(ASCII constant)

由标号标识的地址(addresss pecified by label)

表达式(expression)

④ 注释段(Comment field)：对于这一字段，唯一的要求是用分号起头。注释用来说明该指令在整个程序中的作用。这个字段不产生结果代码，对机器的工作无影响。

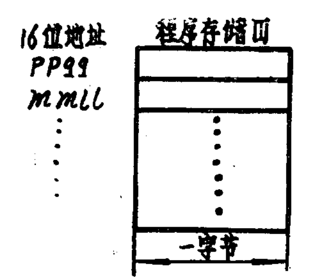


图 4.1 程序存储器的结构

二、指令代码的格式

Z80指令存放在程序存储器(一般用ROM,PROM,EPROM,亦可用RAM)中相继的单元中。每个单元是一个字节，具有唯一的16位地址(见图4.1)。在Z80指令系统中，根据指令内容的不同，一条指令的长度可以是一、二、三或四个字节。第一字节或第一、二字节一定表示操作码。操作数如果存在的话，与操作码一起包含在第一、二字节中，或者单独包含于第三、四字节之中，如图4.2所示的几种情况。

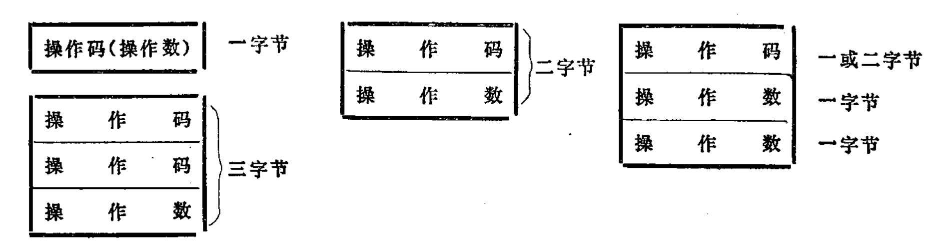


图 4.2 指令代码的格式

三、指令的分类

根据指令所实现的操作，可以将指令分为：

1．数据传送指令：其中包括8位传送指令组；16位传送指令组；交换指令组；数据块传送和查找指令组和输入/输出指令组。

2．数据操作指令：其中包括8位算术和逻辑指令组；16位算术指令组；通用算术指令组和循环与移位指令组。

3．程序控制指令：其中包括转移指令组和子程序调用和返回指令组。

4．CPU控制和位操作指令：其中包括CPU中断控制指令组；CPU其他控制指令组和位操作指令组。

4.2 数据传送指令

数据传送指令是最常用，也是最基本的一类指令。其共同特点是所执行的操作是在CPU寄存器之间或CPU寄存器与存储器之间传送数据；指令必须指明数据传送的源和目的地；且源的内容不因执行传送指令而发生变化。尽管各种传送指令所实现的操作实质上都是相同的，但根据功能可以分为传送、交换、查找、I/O等指令组；根据操作数的长度不同又可分为8位传送指令组，16位传送指令组。下面分别介绍。

一、8位传送指令组

这组指令的一般形式为

LD dst_8 src_8(LoaD source to destination)

所执行的操作是，将源内容传送到目的地。所传送的内容都是8位数据。这个源可以是某个寄存器，令

$$src_8 = r' = A,\ B,\ C,\ D,\ E,\ H,\ L$$

及

$$dst_8 = r = A,\ B,\ C,\ D,\ E,\ H,\ L$$

则有一字指令　　LDr，r′

其代码格式为

0 1 d d d s s s

其中

000——dst或src = B

001——dst或src = C

010——dst或src = D

011——dst或src = E

100——dst或src = H

101——dst或src = L

111——dst或src = A

例如，寄存器的内容H8A_H，记作H = 8A_H，E = 10_H，则执行指令

LDH，E后

$H = 10_H$，$E = 10_H$

式 $H = 10_H$ 中左端的H表示"寄存器 H"，右端的 H 表示 10 为十六进制数。

传送的源可以是一个 8 位数据。因为这个数据就包含在指令之中并直接参加操作，所以叫做立即数。若用 n 表示 0——255 范围内的一字节立即数，则有

LDr，n

表示将立即数 n 传送到寄存器 r，记作

r←n

传送的源或目的地可以是一个指针所指向的存储单元。例如(HL)表示由寄存器对 HL 内容作为指针所指向的存储单元的内容；(IX+d)表示由变址寄存器 IX 的内容与某个位移量 d(displacement)相加形成的指针所指向的存储单元的内容。这样，我们有以下指令

LD r，(HL)，	r←(HL)；
LD (HL)，r，	(HL)←r；
LD r，(IX+d)，	r←(IX+d)；
LD r(IY+d)，	r←(IY+d)
LD (IX+d)，r，	(IX+d)←r；
LD (IY+d)，r，	(IY+d)←r.

有的指令以某个寄存器对的内容为指针，在所指向的存储单元和累加器之间传送数据。这样的指令有

LD A，(BC)，	A←(BC)；
LD A，(DE)，	A←(DE)；
LD (BC)，A，	(BC)←A；
LD (DE)，A，	(DE)←A.

刷新寄存器 R 和中断矢量寄存器 I 只有 8 位，它们与累加器之间可以传送数据

LD A，R，	A←R；
LD A，I，	A←I；
LD R，A，	R←A；
LD I，A，	I←A.

也可以在指令中用一个 16 位数据规定单元的地址，在此单元和累加器之间进行 8 位数据传送。

LD A，(nn)，	A←(nn)；
LD (nn)，A，	(nn)←A.

二、16位传送指令组

这组指令的一般形式为

LD dst_{16}，src_{16}

其中 dst_{16} src_{16} 和是某寄存器对的(16 位)内容或以二字节数值 nn 和 nn+1为地址的相继单元的(16 位)内容。各条指令的操作数(dst_{16} 和 src_{16})的具体含义详见《Z80 袖珍设计手册附表》(以下简称"附表")。这里要指出，有几条指令的操作段包含 8 位和 16 位两种操

作数。这时的操作仍是16位的，分两步完成。例如

LD HL，(nn)

其中　HL——寄存器对HL，16位

(nn)——16位地址nn所指定的存储单元内容，8位

此指令所实现的操作为

H←(nn+1)——将地址为nn+1的单元内容装入H；

L←(nn)——将地址为nn的单元内容装入L.

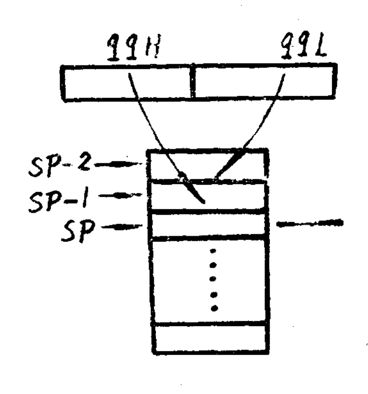


图 4.3　PUSH 指令的执行

在16位传送指令组中，还有6条指令涉及栈操作。先看进栈。源是qq（寄存器对AF，BC，DE，HL中的任意一个），IX或IY。以PUSHqq为例，它所执行的操作，是将寄存器对qq的内容从栈顶压入栈中。其操作如图4.3所示，记作

PUSHqq

(sp－2)←qqL

(sp－1)←qqH

再看出栈指令POPqq。它所执行的操作是将栈顶内容依次弹出，分别送入寄存器对qq的高位和低位中。其操作如图4.4所示，记作

POPqq

qqH←(sp+1)

qqL←(sp)

sp←sp+2

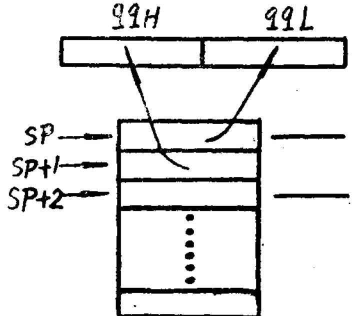


图 4.4　POP 指令的执行

三、指令交换组

这组指令的功能是交换源和目的地的内容，可以简化某些传送操作。尤其是交换主、辅寄存器内容的指令EXX和EX，AF，AF′，在单级中断保护现场时可代替繁冗的栈的操作。交换指令组共有下列六条指令：

指令	**主要用途**
EX.DE，HL(DE←→HL)	交换以便建立指针 (EXchange to set the pointer)
EX AF，A′F′(AF←→A′F′)	交换以便保护现场 (EXchange to save status)
EXX (BC←→D′C′，DE←→D′E′，HL←→H′L′)	
EX (SP)，HL (H←→(SP+1)，L←→(SP))	交换以便保护指针
EX (SP)，IX (IXH←→(SP+1)，IXL←→(SP))	进栈并取出栈顶内容作为新指针 (EXchange to save pointer in stack and set previoue top of stack as new pointer)

$$EX(SP),\ IY\ \begin{pmatrix} IY_H \leftarrow\rightarrow (SP+1) \\ IY_L \leftarrow\rightarrow (SP) \end{pmatrix}$$

四、数据块传送和查找指令组

利用数据块传送指令能将多达65536字节的数据在两个存储器缓冲区之间传送。这两个缓冲区可以位于存储器中的任何部位。在这组指令的操作中，HL寄存器对指向源缓冲区，DE寄存器对指向目的地缓冲区，BC寄存器对作为字节计数，如图4.5所示。

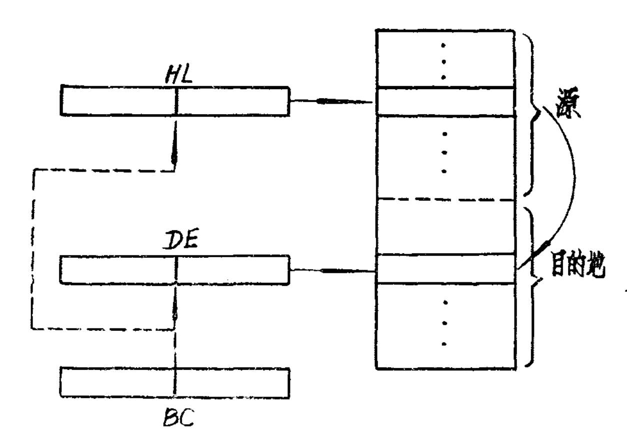


图 4.5 数据报传送指令

数据块传送指令有两种传送方式：

1. 增址型，即每传送一个数据字节后，源/目的地地址增1，数据从低地址向高地址传送；

2. 减址型，即每传送一个数据字节后，源/目的地地址减1，数据从高地址向低地址传送。

两种类型的指令都能再分为单步型和循环型两种。单步型指令只执行一次操作。循环型指令重复同一个操作，直至某个测测条件得到满足为止。现分述如下：

1）增址单步指令LDI。在存储单元间传送数据及目的地和源地址增量(transfer data between memory locations, Increment destination and source addresses)。记作

$$(DE) \leftarrow (HL),\ DE \leftarrow DE+1,\ HL \leftarrow HL+1,\ BC \leftarrow BC-1$$

例如：$BC=0007_H$，$DE=2222_H$，$HL=1111_H$，$(1111_H)=88_H$。执行指令LDI后

$HL=1112_H$，$DE=2223_H$，$(2222H)=88_H$，$(1111_H)=88_H$，$BC=0006_H$

2）增址循环指令LDIR。这条指令与LDI完成相同的操作。只是计数寄存器BC不断自动减1，每次都按新地址传送数据，直至BC=0，操作自动停止。

例如：$HL=1111_H$，$DE=2222_H$，$BC=0003_H$，各单元内容为

$(1111H)=88_H$	$(2222_H)=66_H$
$(1112H)=36_H$	$(2223_H)=59_H$
$(1113H)=A5_H$	$(2224_H)=FC_H$

执行LDIR后，寄存器对和各存储单元内容变为

$HL=1114_H$，$DE=2225_H$，$BC=0000_H$，

以及

$(1111_H)=88_H$　　　　$(2222_H)=88_H$

$(1112_H)=36_H$　　　　$(2223_H)=36_H$

$(1113_H)=A5_H$　　　　$(2224_H)=A5_H$

3）减址单步指令 LDD。在存储单元之间传送数据及目的地和源地址减量（transfer data between memory locations, Decrement destination and source addresses）。记作

$$(DE)\leftarrow(HL),\ DE\leftarrow DE-1,\ HL\leftarrow HL-1,\ BC\leftarrow BC-1$$

4）减址循环指令 LDDR。这条指令与 LDD 相应，计数值 BC 自动减 1，直至 BC＝0 时，操作自动停止。

还应指出，Z80 的数据块传送指令同样适用于动态存储器。在执行这类指令时，动态存储器被不断自动刷新。

当我们对一个大存储器的缓冲区进行搜索，以便找到其中的某个字节时，Z80 数据块查找指令能够提供极大便利。查找指令能使存储器缓冲区逐字节地与累加器内容相比较。“HL 寄存器对”作为缓冲区的存储单元地址指示器，BC 寄存器对作为字节计数。在查找操作中，每一步比较的结果主要反映在 Z 和 P/V 标志中，见图 4.6。

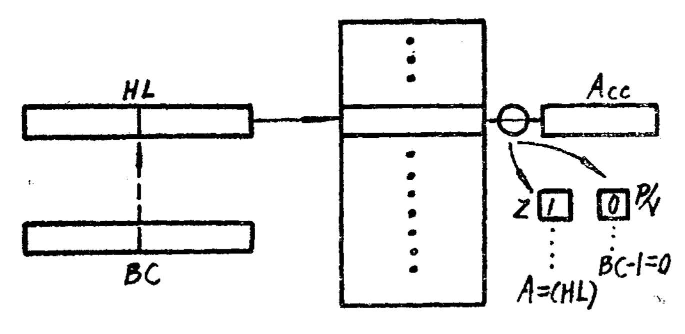


图 4.6　数据块查找指令

与数据块传送指令相仿，这组指令也有增址、减址、单步、循环之分。下面分别介绍。

5）增址单步指令 CPI。比较累加器和存储单元，增量地址，减量字节计数器(Compare accumulator with memory location、Increment address、decrement byte Counter)。记作

$$A\leftarrow(HL)、HL\leftarrow LL+1、$$
$$BC\rightarrow BC-1$$

这条指令对标志的影响如下：

Z——等于 1，若 A＝(HL)；

　　等于0，若A≠(HL)。

P/V——等于 1，若 BC－1≠0

　　等于 0，若 BC－1＝0。

其他标志

S——等于 1，若比较结果为负(A－(HL)<0)；

等于 0，若比较结果为正。

H——等于 1，若第 4 位无错位；

等于 0，若第 4 无位错位；

N——置 1。

CY——不受影响。

6）增址循环指令 CPIR。比较累加器和存储单元，增量地址，减量字节计数器。继续执行直至比较的双方一致或者字节计数等于零为止(ComPare acumulator with memory location, lncrement address, decrement byte counter, continue untit match is found or byte counter is zeRo)。记作

A←(HL)、　　HL←HL+1

BC←BC-1

重复执行直至 A=HL 或 BC=0

例如：　$HL=1111_H$，　$A=F3_H$，　$BC=0004_H$

且若　$(1111H)=52_H$

$(1112H)=00_H$

$(1113H)=F3_H$

$(1114H)=75_H$

$(1115H)=00_H$

执行 CPIR 指令后

$HL=1113_H$，　BC=2，　P/V=1，　Z=1

又如：　$A=3F_H$，　则执行 CPIR 指令后

$HL=1115_H$，　$BC=0000_H$，　P/V=0，　Z=0

减址型查找指令也有两条：CPD 和 CPDR，这里不再赘述。

五、输入输出(I/O)指令组

Z80 向用户提供三组 I/O 指令。

1．标准 8080 输入和输出指令：

1）INA，(n)。输入到累加器(INput to accumulator)。传送数据的源是以 n=0～255作为设备号(device number)所指定的外部设备(I/O 口)，目的地是 CPU 累加器 A。在执行此指令时，操作数 n 放在地址总线的低8位 A_0 至 A_7，以便从最多 256 个口中选择一个口。与此同时，累加器的内容出现在地址总线的高 8 位 A_8 至 A_{15}。然后，所选中 I/O 口的一字节数据被放到数据总线上并写入累加器中。

例如，累加器 $A=23_H$，I/O 口 $(01_H)=7B_H$

执行指令　　IN A，(01_H) 后

$A=7B_H$

2）OUT (n)、A。从累加器输出(OUTput from accum ulator)。传送数据的源是累加器，目的地是设备号 n 所指定的外部设备(I/O 口)。在执行此指令时，操作数 n 放在地址总线的低 8 位 A_0 至 A_7，以便从最多 256 个口中选择一个口，与此同时，累加器的内容出现在地址总线的高 8 位 A_8 至 A_{15}。然后，累加器内容被放到数据总线上并写入指令

操作数 n 所选中的 I/O 口中。

2．寄存器间接寻找输入和输出指令：

1）INr，(C)。输入至寄存器(INput to register)。传送数据的源是以寄存器C内容作为设备号所选中的 I/O 口，目的地是寄存器 r=B，C，D，E，H，L，A。在执行指令时，寄存器 C 的内容放在地址总线的低 8 位 A_0 至 A_7，以便从最多 256 个口中选择一个 I/O口。寄存器B的内容放在地址总线的高 8 位 A_8 至 A_{15}。然后，所选中 I/O 口的一字节数据被放到数据总线上并写入指令操作数r所指定的 CPU 寄存器中。当一段程序中多次访问同一I/O 口时，这种指令使用起来特别方便。

2）OUT (C)，r。从寄存器输出(OUTput from register)。完成操作(C)←r，例如

$$(C)=1F_H, \qquad (H)=AA_H$$

执行指令 OUT(C)，H后设备号为 $1F_H$ 的 I/O 口内容为$(1F_H)=AA_H$。

此指令的详细操作过程类似于上列指令，不再介绍。

3．数据块传送输入和输出指令：

这组指令的操作方式与数据块传送指令相似，只是作为传送一方的源或目的地不是存储器而是 I/O 口。这组指令也有增址、减址、单步、循环之分。下面简要介绍。

1）增址循环输入指令 INIR。输入到存储器并增量指针，直至字节计数器为 0(INput to memory and Increment pointer until byte counter is zero)。记作

$$(HL)\leftarrow(C), \qquad HL\leftarrow HL+1, \qquad B\leftarrow B-1$$

详细过程是：寄存器 C 的内容被放到地址总线的低 8 位 A_0 至 A_7 以便在最多 256 个 I/O 口中选择一个。字节计数器 B 的内容被放到地址总线的高 8 位 A_8 至 A_{15}。然后，所选中 I/O 口的一字节数据放到数据总线上写入 CPU。接着，HL 寄存器对的内容被放到地址总线上，以便将输入的数据字节送入所指向的存储单元。此后，HL 增 1，字节计数器减 1。若减 1 后 B=0，则此指令结束；若 B≠0，则 PC−2，重复执行此指令，如果执行此指令之前将B置为 0 实则根据变补作减法的原理，将输入 256 个字节。

2）减址循环输入指令 INDR。其操作记为

$$(HL)\leftarrow(C), \qquad HL\leftarrow HL-1, \qquad B\leftarrow B-1$$

相应的一对输出指令为 OTIR 和 OTDR，这里不再赘述。

3）增址单步输入指令 INI。它的执行只完成一个操作序列

$$(HL)\leftarrow(C), \qquad HL\leftarrow HL+1, \qquad B\leftarrow B-1$$

此外，也是 IND，OUTI，OUTD 三条指令，详见附表。

4.3 数据操作指令

这类指令主要对 CPU 寄存器中的数据进行算术或逻辑操作，可以分为以下四组：

一、8 位算术和逻辑指令组：实现加(ADD)，带进位加(ADd with Carry)，减(SUBtract)，带进位减(SuBtract with Carry)，“与”(AND)，“或”(OR)，“异”(eXclusive OR)，比较(ComPare)，增量(INCrement)，减量(DECrement)等十一种操作。上述指令的共同特点(INC 和 DEC 指令除外)是；

• 都是针对累加器与某个指定寄存器、存储单元或立即数据之间进行操作的；

• 除了CP指令对累加器没有影响外，其他指令操作的结果均放入累加器中；

• 作为特定操作的结果，将对标志寄存器发生某种影响。

下面介绍各指令。

1）ADD。源字与累加器内容二进制加（binary ADD source word with contents of accumulator）。这条指令因操作数不同具有下列形式：

$$\left.\begin{matrix}\text{ADD}\\ \text{ADC}\end{matrix}\right\}\text{A, r}\qquad \text{r}=\text{A, B, C, D, E, H, L,}$$

所执行的操作是 $A \leftarrow A + r(+CY)$

$$\left.\begin{matrix}\text{ADD}\\ \text{ADC}\end{matrix}\right\}\text{A, nn}=0\sim255(\text{立即数})$$

所执行的操作是 $A \leftarrow A + n(+CY)$

$$\left.\begin{matrix}\text{ADD}\\ \text{ADC}\end{matrix}\right\}\text{A, (HL)}$$

所执行的操作是 $A \leftarrow A + (HL) + (CY)$

$$\left.\begin{matrix}\text{ADD}\\ \text{ADC}\end{matrix}\right\}\text{A,}\ \begin{matrix}(IX+d)\\ (IY+d)\end{matrix}$$

所执行的操作是 $A \leftarrow A + \begin{matrix}(IX+d)\\ (IY+d)\end{matrix}(+CY)$

这里$\begin{matrix}(IX+d)\\ (IY+d)\end{matrix}$是变址寄存器内容$\begin{matrix}IX\\ IY\end{matrix}$作为基地与偏移d相加形成的有效地址所指向的单元。

2）SUB。累加器内容与源字二进制减（binary SUBtract source word from contents of accumulator）。这条指令因操作数不同具有下列形式：

指令 **操作**

$$\text{SUB(SBC)}\left\{\begin{matrix}r\\ n\\ (HL)\\ (IX+d)\\ (IY+d)\end{matrix}\right.\qquad A \longleftarrow A - \left\{\begin{matrix}r\\ n\\ (HL)\\ (IX+d)\\ (IY+d)\end{matrix}\right\}(-CY)$$

3）AND。寄存器、立即数据或存储单元和累加器逻辑"与"（logical AND register, immediate data or memory location with accumulator）。这条指令因操作数不同具有下列形式：

指令 **操作**

$$\text{AND}\left\{\begin{matrix}r\\ n\\ (HL)\\ (IX+d)\\ (IY+d)\end{matrix}\right.\qquad A \longleftarrow A \wedge \left\{\begin{matrix}r\\ n\\ (HL)\\ (IX+d)\\ (IY+d)\end{matrix}\right.$$

4）OR。寄存器、立即数据或存储单元和累加器逻辑"或"（logical OR register,

immediate data or memory location with accumulator)。这条指令因操作数不同具有下列形式：

$$\text{OR}\begin{cases} r \\ n \\ (HL) \\ (IX+d) \\ (IY+d) \end{cases} \qquad A \leftarrow A \wedge \begin{cases} r \\ n \\ (HL) \\ (IX+d) \\ (IY+d) \end{cases}$$

（左：指令；右：操作）

5）XOR。寄存器，立即数据或存储单元和累加器逻辑"异"(logIcal exclusive OR register, immediate data or memory location with accumulator)。这条指令因操作数不同具有下列形式：

$$\text{XOR}\begin{cases} r \\ n \\ (HL) \\ (IX\to d) \\ (IY+d) \end{cases} \qquad A \leftarrow A \oplus \begin{cases} r \\ n \\ (HL) \\ (IX+d) \\ (IY+d) \end{cases}$$

（左：指令；右：操作）

6）CP。寄存器、立即数据或存储单元和累加器内容相比较(Comparere rgister, immediate data or memory location with contents of accumulator)。这条指令因操作数不同具有下列形式：

$$\text{CP}\begin{cases} r \\ n \\ (HL) \\ (IX+d) \\ (IY+d) \end{cases} \qquad A - \begin{cases} r \\ n \\ (HL) \\ (IX+d) \\ (IY+d) \end{cases}$$

（左：指令；右：操作）

在执行此指令时，累加器与源数据不变，比较结果反映在标志位。例如

$$(IX+d) = A0_H,\ A = E3_H$$

执行 CP(IX+d)后，各标志位根据运算结果受到不同影响，如图 4.7 所示。

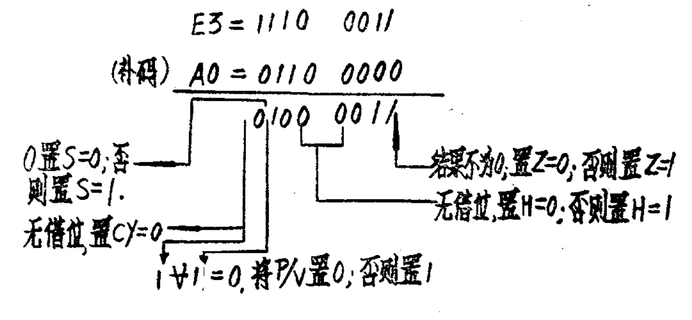


图 4.7 CP 指令举例

7）INC。增量寄存器或存储单元(INCrement register or memory location)。这条

指令因操作数不同具有下列形式：

指令	操作
INC r	$r \leftarrow r+1$
INC (HL)	$(HL) \leftarrow (HL)+1$
INC (IX+d)	$(IX+d) \leftarrow (IX+d)+1$
INC (IX+d)	$(IY+d) \leftarrow (IY+d)+1$

S，Z，H，P/V 标传将受到影响，详见附表。

8）DEC。减量寄存器或存储单元(DECrement register ormemory location)。其操作数与 INC 指令的相同，只是完成"-1"操作，详见附表。

二、16位算术指令组：这组指令一共三种，加、减法指令的操作是在 HL 寄存器对与某一寄存器之间进行的。增减量指令是针对某个寄存器对进行的。

1）ADD HL，ss。HL 与寄存器对相加(ADD register toH and L)。记作

$$HL \leftarrow HL+ss$$

其中 ss=BC，DE，HL，SP

2）ADC HL，ss。HL 寄存器对带进位相加(ADd register pair with Carry to H and L)。记作

$$HL \leftarrow HL+ss+CY$$

3)SBC HL，ss。HL 与寄存器对带进位减(SuBtract register pair with Carry to H and L)。记作

$$HL \leftarrow HL-ss-CY$$

4）INC ss。增量指定寄存器对的内容(INCrement contents ofspecified register pair)。记作

$$ss \leftarrow ss+1$$

5）DECss。减量指定寄存器对的内容(DECrement contents of specified register pair)。记作

$$ss \leftarrow ss-1$$

三、通用算术指令组：这组指令的操作对象是 PSW，在某些算术运算中，它们起重要辅助作用。

1）DAA。十进制调整累加器(Decimal Adjust Accumulator)。将二进制加法自动调整成 BCD 加法。这条指令利用进位 CY 标志和半进位 H 标志，使 BCD 加法得到正确结果。

在 Z80 的汇编语言程序中，利用 DAA 指令与算术运算指令组成十进制运算复合指令组 ADD DAA，ADC DAA，INC DAA，SUB DAA，SBC DAA，DEC DAA，AEC DAA 等效于对 BCD 的源进行运算，产生 BCD 的结果。例如

$A=39_H$，　　　　$B=47_H$，

执行 ADD B

DAA

后，$A=86_H$，而不是 $A=80_H$。

2）CPL。累加器变反(ComPLement the accumulator)。记作

$$A \leftarrow \overline{A}$$

3）NEG。累加器内容变负(NEGate contents of accumulator)。它等效于取累加器内容的补码，即

$$A \leftarrow 0 - A$$

或写成

$$A \leftarrow \overline{A} + 1$$

4）CCF。进位标志变反(Complement Carry Flag)。记作

$$CY \leftarrow \overline{CY}$$

5）SCF．置位进位标志(Set Carry Flag)。记作

$$CY \leftarrow 1$$

四、循环移位和移位指令组：循环移位指令使寄存器形成环形工作方式。其最高位直接与最低位相连。原始数据的各个位保持不变，唯一发生变化的是各个位的位置以及进位标志的内容。这类指令多用于实现乘、除法和计数等程序中。

移位指令实现普通的移位操作。利用移位操作，可以使数据在串行和并行两种形式之间进行转换，实现对数据的定标和规格化；实现对数据的拼装和分离；实现乘、除法比较位组合格式等。

Z80 向用户提供六种基本类型的九组循环移位指令。简要介绍如下：

1）RLC。寄存器、累加器或存储单元向左循环移位。(Rotate register, accumulator or memory location Left Circular)。这条指令因操作数不同具有下列形式：

RLC { r, (HL) }——二字节

RLC { (IX+d), (IY+d(}——四字节

RLCA——————一字节

所执行的操作如图 4.8 所示。第 0 位移入第 1 位；原来的第 1 位移入第 2 位，并依此类推。第 7 位移入进位标志 CY 和第 0 位。

2）RRC。寄存器、累加器或存储单元向右循环移位(Rotate register, accumulator or memory location Right Circular)。其操作数与 RLC 指令相同，所执行的操作如图 4.9 所示。

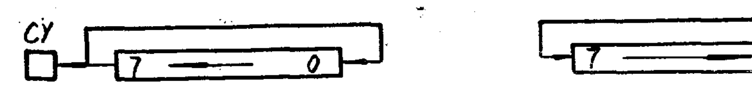


图 4.8 RLC 指令的操作　　　图 4.9 RRC指令的操作

3）RL。寄存器、累加器或存储单元通过进位位向左循环移位(Rotate register, accumulator or memory location Left thru carry)。这条指令因操作数不同具有下列形式：

RL { r, (HL) }——二字节

RL { (IY+d), (IY+d) }——四字节

RLA——————一字节

所执行的操作如图 4.10 所示。

4) RR。寄存器、累加器或存储单元通过进位位向右循环位(Rotate register, accumulator or memory location Right thru carry)。其操作数与 RL 指令相同，所执行操作如图 4.11 所示。

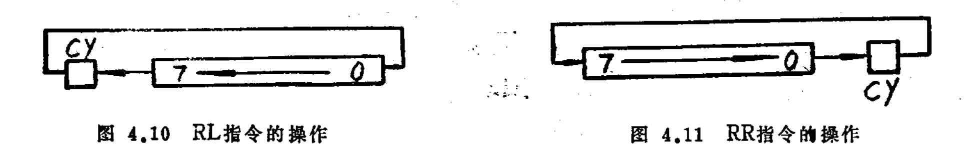


图 4.10 RL指令的操作　　　　图 4.11 RR指令的操作

5) RLD。一个二——十进制数字在累加器和存储单元之间向右循环位(Rotate one BCD digit Leftbetween the accumulator and memory location)。所执行的操作如图 4.12 所示。存储单元(HL)的低 4 位内容移入同一单元的高 4 位(图中①)。该高 4 位原先的内容移入累加器的低 4 位(图中②)。累加器低 4 位原先的内容移入(HL)的低4位(图中的③)。累加器的高4位不受影响。

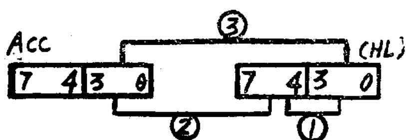


图 4.12 RDL指令的作操

6) RRD。一个二——十进制数字在累加器和存贮单元之间向右循环移位(Rotate one BCD digit Right between the accumulator and memory location)。所执行的操作如图 4.13 所示。

7) SRL。寄存器或存贮单元向右逻辑移位(Shift register or memory location Right Logical)。这条指令因操作数不同具有下列形式：

$$
\mathrm{SRL}\begin{cases} \mathrm{r} \\ (\mathrm{HL}) \\ (\mathrm{IX}+\mathrm{d}) \\ (\mathrm{IY}+\mathrm{d}) \end{cases}
$$

所执行的操作如图4.14所示。数据向左移位，留下的空位都被清零。

8) SLA。寄存器或存贮单元向左算术移位(Shif register or memory location Left Arithmetic)。此指令的操作数与 SRL 指令相同，所执行的操作如图 4.15 所示。可见，这实质上是一条向左的逻辑移位指令。

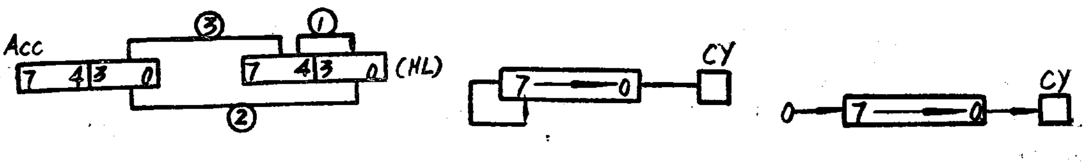


图 4.13 RRD指令的操作　　图 4.14 SRL指令的操作　　图 4.15 SLA指令的操作

9) SRA。寄存器或存贮单元向右算术移位(Shift register or memory location Arithmetic)。此指令的操作数与 SLA 指令相同，其操作如图 4.16 所示。在移位时保留数据最高位(符号位)，并将其依次传送到下一位。这种操作也叫做符号延展(Sign extension)，如图 4.17。

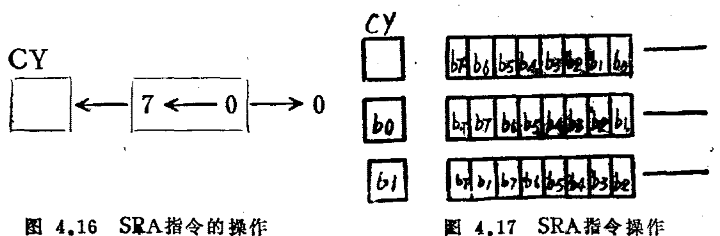


图 4.16 SRA指令的操作　　　　图 4.17 SRA指令操作

4.4 程序控制指令

这类指令控制和改变程序的正常进程。包括:

1. 无条件转移指令(unconditional Jump)。该指令通过对程序计数器的操作。使正在执行的指令序列转向新的地址继续执行下去。

2. 条件转移指令(JumP uncondition)。该指令使 CPU 能重复执行一个指令序列,测试字符,识别错误,检查外部设备的状态等。条件转移指令是使 CPU 能够进行判定的主要手段。可以说 CPU 的智能很大程度上依赖于这条指令。

3. 子程序操作指令。它也是实现程序的转移。但它与普通转移指令的区别在于执行所转移的程序后还能返回到原来的断点处,继续执行原来的程序。下面介绍各类程序控制指令。

一、转移指令组

转移指令又可分为以下四组:

1. 标准 Z80 转移指令。除所用助记符不同外,其功能完全雷同于 8080A 的转移指令。

1) JP $\begin{matrix}nn\\label\end{matrix}$。转移到由操作数所指定的指令。(JumP to the instruction identified in the operand)。这是三字节指令,第一字节规定操作码;二、三字节规定操作数,它可以是一个 16 位地址 nn 也可以是一个表示 16 位地址的标号。这条指令所执行的操作为

$$PC \leftarrow (第三字节)(第二字节)$$

使程序转到操作数所指定的地址。

2) JP cc,$\begin{matrix}nn\\label\end{matrix}$。如果条件满足,则转移到操作数所指定的地址(JumP to address identified in the operand if condition is satisfied)。

cc 是条件,包括:

cc＝NZ——非零(NonZero),　　标志Z＝0
cc＝Z ——零(Zero),　　标志Z＝1
cc＝NC——没有进位(NonCarry),　　标志CY＝0
cc＝C ——有进位(Carry),　　标志CY＝1
cc＝PO——奇偶性奇(Parity Odd),　　标志P/V＝0
cc, PE——奇偶性偶(Parity Even),　　标志P/V＝1
c＝P ——符号为正(sign Positive),　　标志S＝0

cc = N—— 符号为负(sign Negative)，　　　　标志S = 1

2．变址转移。

$$JP\begin{cases}(HL)\\(IX)\\(IY)\end{cases}$$

。转移到由16位寄存器内容所指定的地址(JumP to address specified by contents of 16 bit register。记作

$$PC\begin{cases}HL\\IX\\IY\end{cases}$$

寄存器对HL，IX，IY都是用来贮存指针的。又由于它们可以进行+1/-1操作，因此利用它们的内容作为可变转移地址十分便利。

3．二字节转移。

三字节转移指令多用于"长距离"转移。如果转移的目标就在现行指令附近，一般采用相对于现行指令的二字节转移指令，其中也包括无条件和有条件转移两种。

1) JR e-2。相对于程序计数器现在的内容转移(Jump Relative to present contents of program counter)。操作数中的e是偏移。这里要注意，偏移e是从相对转移指令本身的第一字节(操作码)开始算起的，如图4.18所示。转移，范围是从$(e-2)_{min}+2=-128+2=-126$个字节到$(e-2)_{max}+2=+127+2=129$个字节。

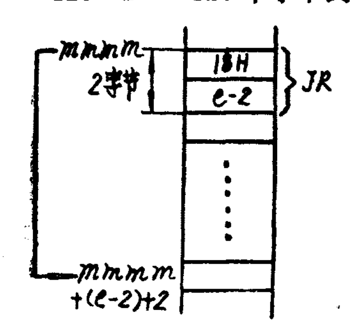


图 4.18　JR指令的操作

2) $JR\begin{matrix}C\\NC\\Z,\\NZ\end{matrix}$ e-2。若操作数中所规定的条件得到满足，相对于程序计数器现在的内容转移(Jump Relative to present contents of program counter if conditions specified in the operand are satisfied)。相对转移的条件是C(CY=1)，NC(CY=0)，Z(Z=1)和NZ=0。

4.循环控制转移指令。

在实际问题中，往往需要多次重复执行一段程序，直至某一条件得到满足为止。这一条件可以由一个预置的计数值逐次减1实现。当计数值为0时，条件得到满足，程序循环结束。为了便于实现这类操作，Z80提供一条专用的循环控制转移指令：

DJNZ e-2。寄存器B减1，若寄存器B不为零，相对于程序计数器现在的内容转移(Decrement register B,Jump relative to present contents of program counter if register B Not Zero)。

DJNZ的典型使用方式如图4.19所示。

二、子程序调用和返回指令组

调用是一种转移，它将程序计数器原先的内容保护起来，以便在执行完所调用的子程序之后，从断点返回原来的程序流程。一般情况下，调用和返回指令是成对编排的，也分为无条件和条件调用/返回两类。下面简要介绍。

1) CALL $\genfrac{}{}{0pt}{}{\text{nn}}{\text{label}}$。调用以 nn 或标号为起始地址的子程序(CALL the subroutine entered with nn or label)。记作

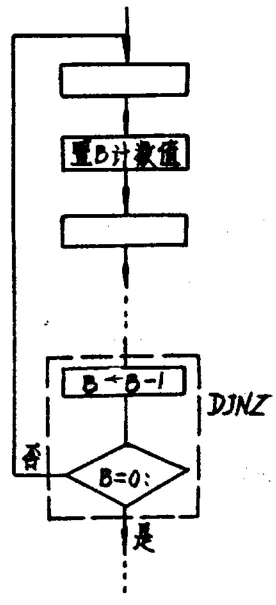


图 4.19 DJNZ 指令的操作

$$(SP-1) \leftarrow PCH$$
$$(SP-2) \leftarrow PCL$$
$$PC \leftarrow nn$$

此指令将 PC 的当前内容压入栈内，然后将 CALL 指令的操作数段所给出的子程序入口地址装入 PC，以便转去执行子程序。

3. RET。从子程序返去(RETurn from subroutine)。记作

$$PC_L \leftarrow (SP)$$
$$PC_H \leftarrow (SP+1)$$
$$SP \leftarrow (SP+2)$$

此指令将原先进栈的断点地址弹回程序计数器而返回原程序流程。

调用和返回指令也可按一定测试条件进行。相应的指令为 CALL cc, nn 和 RET，详见附表。以外，还有一条特殊的一字节调用指令，常常由外部设备软件来提供，即

RST p。重新启动(ReStarT)。它是一个一字节子程序调用指令，最常用于中断的场合，其结果代码为

RST p: 11 XXX 11

t	p
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

PC | 0 0 0 0 0 0 0 0 0 0 X X X 0 0 0 |

t 使程序计数器的新内容变为

能够形成八个中断服务程序入口地址。

执行 RST p 指令时，先将 PC 保护进栈，然后将 p 装入 PC_L, PC_H 为 00，转向中断服务程序。

4.5 CPU 控制和位操作指令

一、CPU 中断控制指令组

中断的目的是使外部设备能够以一定的方式暂停 CPU 的操作，并迫使 CPU 执行一个服务子程序。待子程序完成后，CPU 就返回原先的程序流程。Z80 提供三组中断控制指令，现分述如下：

1．中断允许/禁止。Z80 具有两种中断输入，软件可屏蔽中断 INT (maskablc INTerrupt)和不可屏蔽中断 NMI (Non-Maskable Interrupt)。NMI 是为那些必须要求响应的很重要的功能服务的。INT 则能够由程序员有选择地允许或禁止。其控制是通过对中断允许触发器 IFF(Interrupt Flip-Flop)的操作实现的。在 Z80—CPU 中，有两个这样的触发器 IFF1 和 IFF2。IFF1 的状态用来禁止中断，而 IFF2 用来暂存 IFF1 的内容。

1) EI。允许(开)中断(Enable Interrupt)。此指令会使 IFF1 和 IFF2 都进入置位状态(态状1)。当 CPU 接受中断后，IFF1 和 IFF2 自动复位，从而禁止后继的中断申请，直至程序中出现一个新的 EI 指令为止。

2) DI。禁止(关)中断(Disable Interrupt)。当执行此指令后，可屏蔽中回请求被禁止。I/O 设备向 CPU发出的$\overline{\text{INT}}$输入无效。上述状态一直维持到出现 EI 指令为止。

2．中断返回。

1) RETN。从不可屏蔽中断返回(RETurn from Non-maskable interrupt)。用于不可屏蔽中断服务程序结束时，此指令执行无条件返回，其功能与 RET 指令相同。就是说，先前进栈的 PC 值从栈中弹出，然后 SP←SP＋2。另外，此指令恢复中断允许逻辑，即 IFF2 的状态被传送到 IFF1，使之恢复到接受 NMI 之前所具有的中断状态。

2) RETI。从中断返回(RETurn from Interrupt)。此指令与RET指令的功能完全相同，只是 Z80 外部接口器件来识别此指令，并用它表示现行中断服务程序已经执行完毕。

3．中断方式选择。可屏蔽中断具有方式 0，1，2 三种方式：

1) 方式 0 (MODE0)。执行"IM0。中断方式 0 (Interrupt Mode0)"指令。进入方式 0 的准备条件是：$\overline{\text{RESET}}$ 信号已加至 CPU，使后者完成初始化或 IM0 指令已执行完毕；IFF1 处于置位状态，即已开中断，这时如果没有总线请求或不可屏蔽中断请求，则此中断请求将被接受。CPU 将发出一个 $\overline{M_1}$ 和 $\overline{\text{IORQ}}$ 信号，然后进入等待状态。

请求中断的外部设备应能识别 $\overline{M_1}$ 和 $\overline{\text{IORQ}}$ 的同时出现，并将一条指令放在数据总线上去。一般情况下，所放的是 RST 或 CALL 指令。这两条指令都自动将 PC 保护进栈，然后转到中断服务程序的入口。

一旦中断开始，所有后继的中断都被禁止，IFF1 和 IFF2 自动复位。这时，程序中应安排一条 EI 指令，以便允许优先级较高的其他设备发出中断请求。在中断服务程序结尾，应安排 RETI 指令，使中断结束返回原先的程序流程。方式 0 的操作如图 4.20 所示。

2）方式1（MODE1）。执行“IM1。中断方式 1(Interrupt Mode1)”指令。CPU 响应中断时自动执行一条 RST 指令，转向单元 0038H。可见，这种中断方式所需外部硬件最少。中断方式1的操作见图 4.21。

3）方式2（MODE2）。执行“IM2。中断方式2（Interrupt Mode2)”指令。它也叫做矢量中断方式。中断矢量的低8位由请求中断的外部设备提供。方式2的操作见图 4.22。

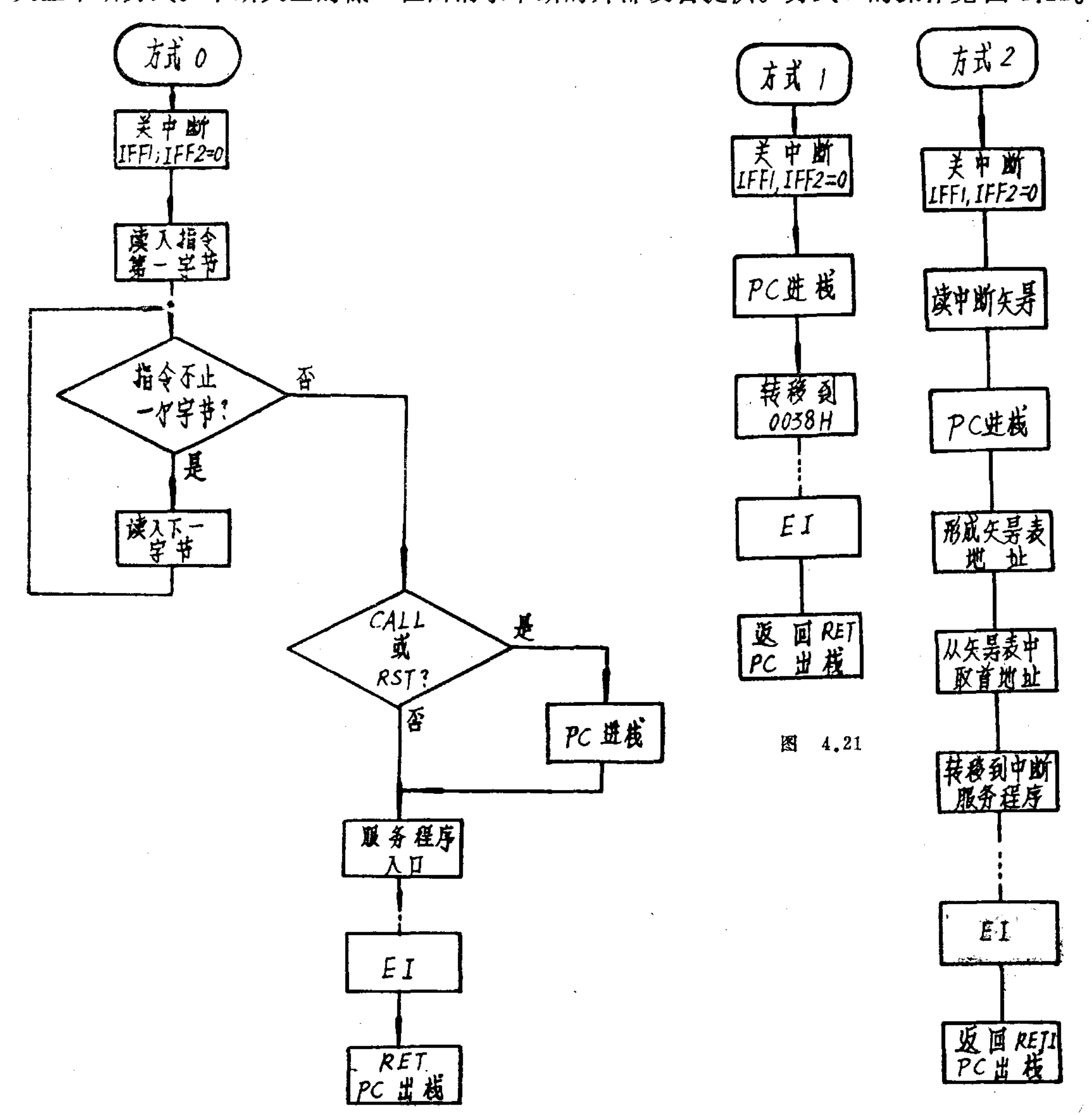


图 4.21

图 4.20 中断方式0的操作

图 4.22 中断方式2的操作

二、CPU 其他控制指令组

1）NOP。不操作(No Operation)。执行此指令时除程序计数器增1和动态存贮器刷新外，CPU 不实现任何操作。

2）HALT。暂停(HALT)。它使 CPU 停止操作，重复执行 NOP 指令，直至接到中断请求或复位命令后才重新开始进入程序流程。

三、位操作指令组

这类指令使我们能对寄存器或存贮单元中的指定位进行测试或置位、复位操作。例如在信号处理的应用中，一个单独信号往往作为一个二进制位的实体存在，这时位操作就具有特别重要的意义。Z80 的位操作指令有以下几组：

1) BIT b、r

b，(HL)

b，(IX+b)

b，(IY+d)

测试寄存器或存贮单元中的一位，结果放在零标志中(test BIT in register or memory location，the result is put in zero flag)。指令中第二个操作数指定被测试位所在的源字节，第一个操作数指定测试位的位号，它的代码为3位。

2) SET b，r

b，(HL)

b，(IX+d)

b，(IY+d)

置位寄存器或存储单元中的指定位b (SET bit b of register or memory location) 记作

$r_b \leftarrow 1$

$(HL)_b \leftarrow 1$

$(IX+d)_b \leftarrow 1$

$(IY+d)_b \leftarrow 1$

3) RES b，r

b，(HL)

b，(IX+d)

b，(IY+d)

复位寄存器或存贮单元中的指定位b (RESet bit b of register or memory location)。

第五章　并行输入/输出接口芯片 Z80 PIO

5.1 概　　述

一个微型计算机，除 CPU 和 RAM、ROM 外，尚需要输入/输出(I/O)芯片，才能使系统正式进行操作。这些 I/O 芯片可分为两大类：

1. 通用 I/O 芯片。其中主要有并行 I/O、串行 I/O、DMA 可编程中断控制器、计数/定时器等等。

2. 专用 I/O 芯片。其中主要有软磁盘控制器、CRT 控制器、键盘接口、键盘/显示接口、数据编码等等。

本书在第五、六两章中只介绍 Z80 系列中最常用的两个芯片：Z80-PIO 和 Z80-CTC。

Z80-PIO(以下简称为 PIO)是一个可以用程序来变更其工作方式的器件，它具有16根输入/输出(又称为 I/O)线。这些 I/O 线可分成两个8位的 I/O 口，每个 I/O 口配有二根联络线(handshaking)，用以控制数据的传送。

PIO 能为外部设备与 Z80-CPU 之间提供一个 TTL 兼容的接口。利用 CPU 的指令变更 PIO 的工作方式，从而能与各种各样的外部设备——大多数的键盘、纸带读入机和凿孔机、打印机、PROM 写入器等——相接而不需其他外加电路。

当使用 PIO 时，外部设备与 CPU 之间的数据传送均在 IM2(方式 2)中断控制下实现。

5.2 PIO 的方框图及引脚

PIO 的方框图示于图 5.1。每个口的 I/O 逻辑是由六个寄存器和"联络"控制逻辑所组成，如图 5.2 所示。六个寄存器为：8 位数据输入寄存器；8 位数据输出寄存器；2 位方式控制寄存器；8 位屏蔽寄存器；8 位 I/O 选择寄存器；2 位屏蔽控制寄存器。

前两种寄存器统称为数据寄存器，后四种寄存器统称为控制寄存器。这些控制寄存器的作用将在 5.3 节中叙述。

PIO 的引脚布置如图 5.3 所示。这些引脚的作用如下：

1. D7—D0 为数据总线(双向，三态)。这个总线用来在 CPU 和 PIO 之间传送数据和命令。

2. $\overline{CE}$ 为芯片允许(输入，低电平有效)。只有当此信号为低电平，才允许 CPU 访问该 PIO 芯片。

3. B/A 为选择接口 A 或口 B(输入)。当此信号为低电平时，选中口 A；当为高电平时，选中口 B。通常将此引脚接至 CPU 地址总线的 A0 位。

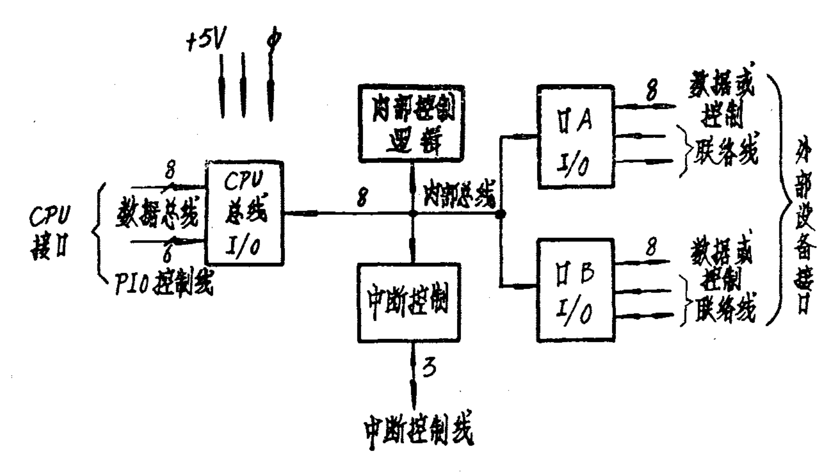


图 5.1 PIO 的框图

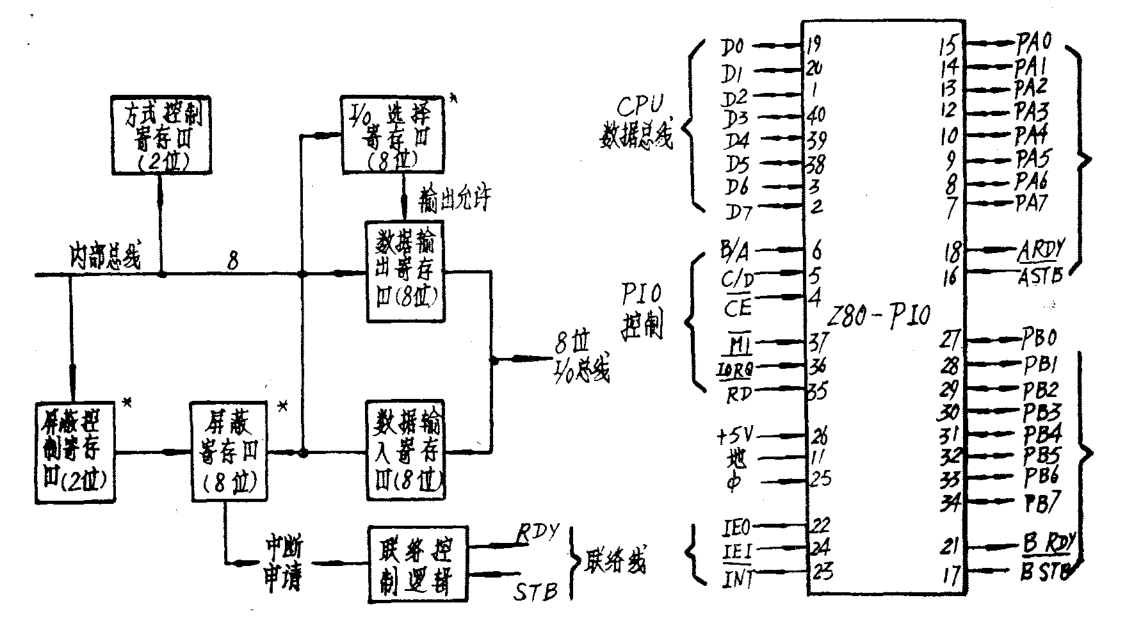


图 5.2 口的 I/O 框图

图 5.3 PIO 的引脚布置图

4．C/D 为选择数据寄存器或控制寄存器(输入)。当该信号为低电平时，访问数据寄存器，即数据总线被用来传送数据；当该信号为高电平时，访问控制寄存器，即 CPU 通过数据总线向 PIO 写入的信息是一个命令。通常将此引脚接至 CPU 地址总线的A1位。

从以上三个信号的含义，可将 PIO 的选择逻辑归纳如表 5.1 所示。

表 5.1 中×表示可为任意值。80_H 表示为十六进制数 80。如果地址总线 A7—A2＝100000 时，$\overline{CE}$ 被译码成逻辑 0。如果 A7—A2 为其他值时，$\overline{CE}=1$。在这种情况下，PIO 占有四个外部设备地址 80_H，81_H，82_H，83_H。

5．$\overline{M1}$ 为 CPU 的取指令机器周期信号(输入、低电平有效)。

6．$\overline{IORQ}$ 为 CPU 的 I/O 请求信号(输入，低电平有效)。

7．$\overline{RD}$为 CPU 的读周期状态(输入，低电平有效)。

上述三种控制信号的作用如表 5.2 所示：

表 5.1　　PIO 的选择逻辑

引脚			选中单元	单元地址
$\overline{CE}$(*)	B/A(A0)	C/D(A1)		
0($A_7-A_2=100000$)	0	0	口A的数据寄存器	80_H
0($A_7-A_2=100000$)	0	1	口A的控制寄存器	82_H
0($A_7-A_2=100000$)	1	0	口B的数据寄存器	81_H
0($A_7-A_2=100000$)	1	1	口B的控制寄存器	83_H
1($A_7-A_2\neq100099$)	×	×	芯片被选中	

(*) CJ—801 是按此地址连接的。不同的微型机系统或不同的 PIO 芯片，A_7-A_2 可为不同值。

表 5.2　　PIO 控制信号的作用

引脚			功能解释
$\overline{M1}$	$\overline{IORQ}$	$\overline{RD}$	
0	0	0	没有作用
0	0	1	中断响应
0	1	0	检查中断服务程序是否结束
0	1	1	复　位
1	0	0	CPU 从 PIO 读出
1	0	1	从 CPU 写入 PIO
1	1	0	没有作用
1	1	1	没有作用

8．IEI 为中断允许输入信号(输入，高电平有效)。当用了一个以上的中断源器件时，本信号被用来构成优先权中断链。若本引脚为高电平，表示 CPU 目前没有为优先权比本芯片更高的其他器件服务。此时允许芯片向 CPU 请求中断。

9．IEO 为中断允许输出信号(输出，高电平有效)。只有 IEI 为高电平，且 CPU 没有为本芯片的中断服务时，本信号才为高电平。若 CPU 为本芯片或中断优先权更高的芯片的中断服务时，本信号均为低电平，从而阻止优先权较低的器件发出中断请求。

10．$\overline{INT}$ 为中断请求(输出，漏极开路，低电平有效)。当 PIO 向 CPU 发出中断请求时，$\overline{INT}$ 有效。

下面便介绍链形中断的处理过程。图 5.4 表示了典型的嵌套中断的序列。在此序列中，口 2A 首先发中断请求(在同一芯片中，口 A 的中断优先权高于口 B)并被接受。当口 2A 正在被处理中，一个优先权更高的口 1B 发出中断请求，并被接受。接着对优先权较高的口 1B 的服务程序执行完毕，并且执行一条 RETI 指令，通知此口这一服务程序已结束，口 1B 的 IEO 端的电位变高。接着完成优先权较低的 2A 口服务程序。

11．PA7-PA0 为口 A 的总线(双向，三态)。用来在口 A 和外部设备之间传送数据和控制信息。

12．$\overline{ASTB}$ 为(来自外部设备的)口 A 选通脉冲(输入，低电平有效位)。此信号的作用与口 A 的工作方式有关。将在下节介绍。

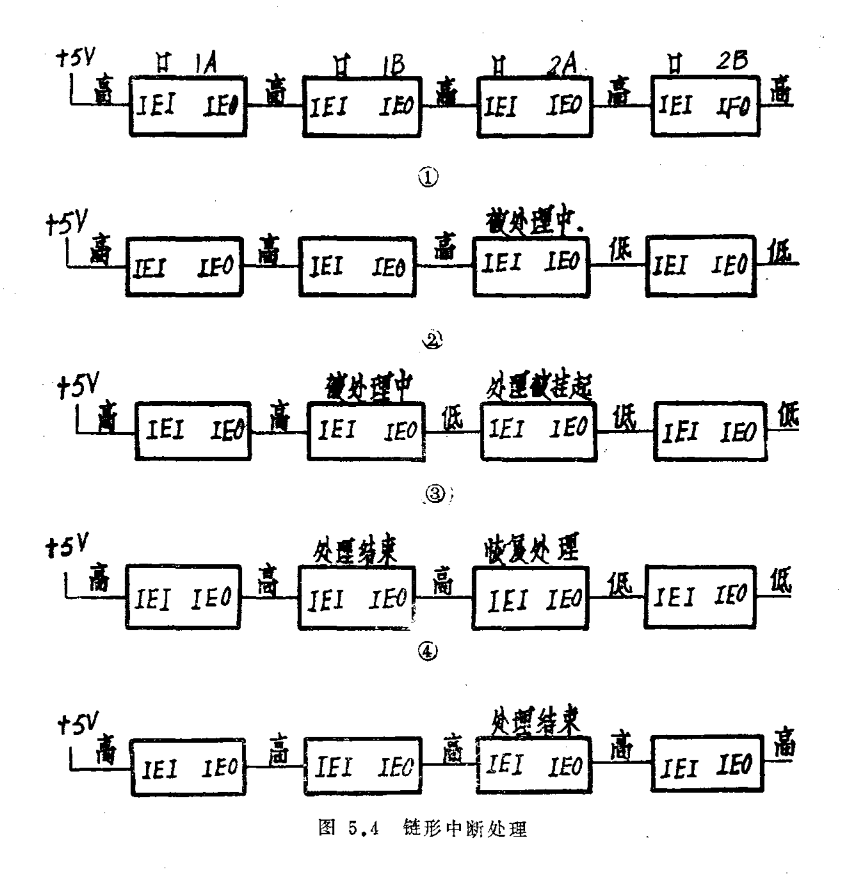


图 5.4 链形中断处理

13. ARDY 为寄存器 A 待命(输出，高电平有效)。此信号的作用与口 A 的工作方式有关将在下节介绍。

14. PB7-PB0 为口 B 的总线(双向，三态)。用来在口 B 和外部设备之间传送数据和控制信息。口 B 总线能在 1.5 伏下提供 1.5 毫安电流，以便驱动复合晶体管。

15. $\overline{\text{BSTB}}$ 为来自外部设备的口 B 选通脉冲(输入，低电平有效)。此信号的作用与口 B 和口 A 的工作方式有关，将在下节介绍。

16. BRDY 为寄存器 B 待命(输出，高电平有效)。此信号的作用与口 B 和口 A 的工作方式有关，将在下节介绍。

17. 其他 ϕ 为时钟；+5V 为电源；GND 为地。

5.3 PIO 的操作说明

每片 PIO 占有四个外部设备地址，即有四个可寻址单元能被访问，如图 5.5 所示。图中的设备地址采用 CJ801 单板计算机中所使用的具体值。

PIO 有两个口：口 A 和口 B。每个口有两个可寻址单元，一是控制寄存器，另一是数据寄存器，写入控制寄存器的控制字有六种，其中有三种仅用于工作方式 3，将在方式 3 中介绍。下面介绍另外三种(也是更重要的)控制字的格式及含义。由于这些控制字都是写入同一地址的控制寄存器，因而需要规定一些特征来区别这些控制字。

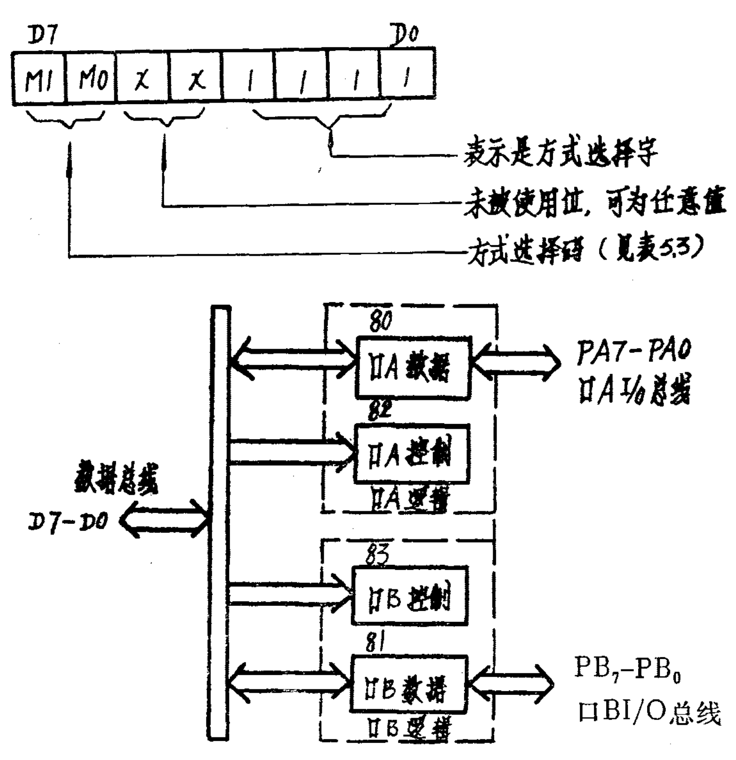


图 5.5 PIO 有四个可寻址单元

1、方式选择字

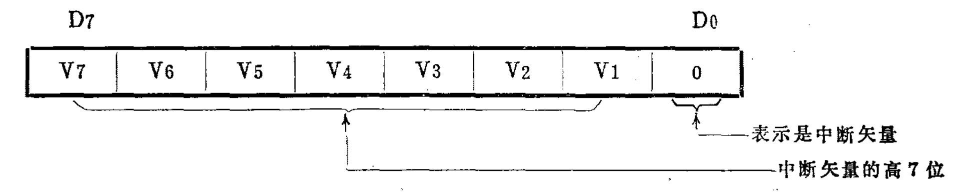
PIO 有四种操作方式，如表 5.3 所示。具体操作方式选择字中的 M1M0 来决定。

表 5.3　　PIO 操作方式

M1	M0	操作方式	说明
0	0	方式 0	带联络的输出
0	1	方式 1	带联络的输入
1	0	方式 2	带联络的双向 I/O*
1	1	方式 3	位控操作方式

*只有口 A 能工作在方式 2，此时口 B 的联络线被口 A 所占用，因而口 B 只能工作在方式 3。详细解释见后。

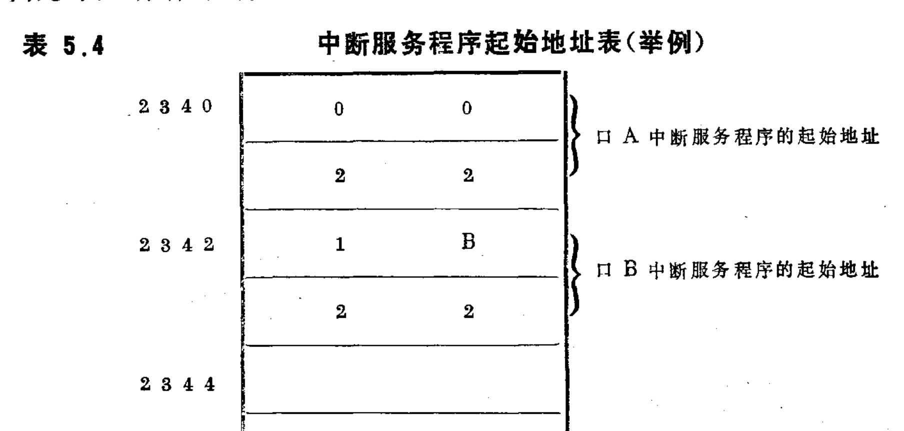
2、中断矢量



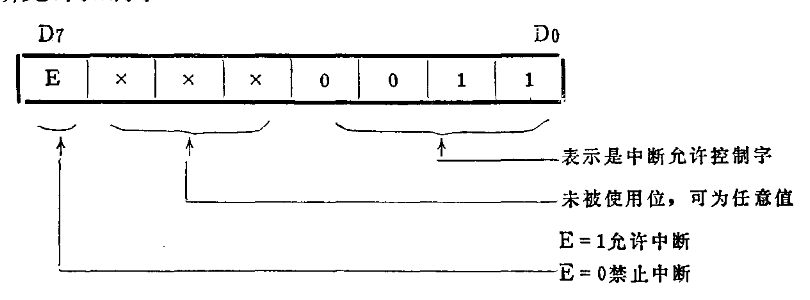
假设 CPU 中断矢量寄存器 I＝23H，又设口 A 的 V7－V1＝0100000，则 PIO 的中断矢量＝40H，两者合成一个完整的中断矢量 2340H。如表 5.4 所示，由 2340 和 2341 单元中找到中断服务程序的起始地址 2200H。因为中断服务程序的起始地址占用两个单元。

因而中断矢量可以按排成都是偶数，如 2340_H，2342_H，2344_H 等，即它的最低位必然为0。由此可以与其他控制字相区别。

表 5.4　　中断服务程序起始地址表(举例)



3. 中断允许控制字



现分别介绍四种操作方式

一、方式 0—输出。

以口 A 为例，结合下面的初始化程序，说明 PIO 如何被设定成这种工作方式并进行工作。

```
LD      A，23H        设定中断矢量
LD      I、A
LD      A、40H
OUT     (82)，A
LD      A、0FH        设定为工作方式 0
OUT     (82)、A
LD      A、83H        允许 PIO 请求中断
OUT     (82)，A
IM      2             设置中断方式 IM2
EI                    开中断
LD      A，(HL)
OUT     (80)，A       向口A输出一个数据
```

其过程如下：

1．设定中断矢量为2340H。

2．设定口A为操作方式0。

3．对 PIO 口A开中断，也就是使其中断允许触发器置位。

4．设定 CPU 的中断方式为 IM2，并开 CPU 的中断。

5．CPU 执行一条输出指令 OUT(80)，A 即开始方式 0 的输出周期，如图5.6所示。CPU 执行输出指令期间，对 PIO 产生一 $\overline{WR^*}$ 脉冲，并用它将累加器 A 中的数据锁存在口 A 的数据输出寄存器中。之后 $\overline{WR^*}$ 电平变高，在下一个 φ 下降沿使 ARDY 电平变

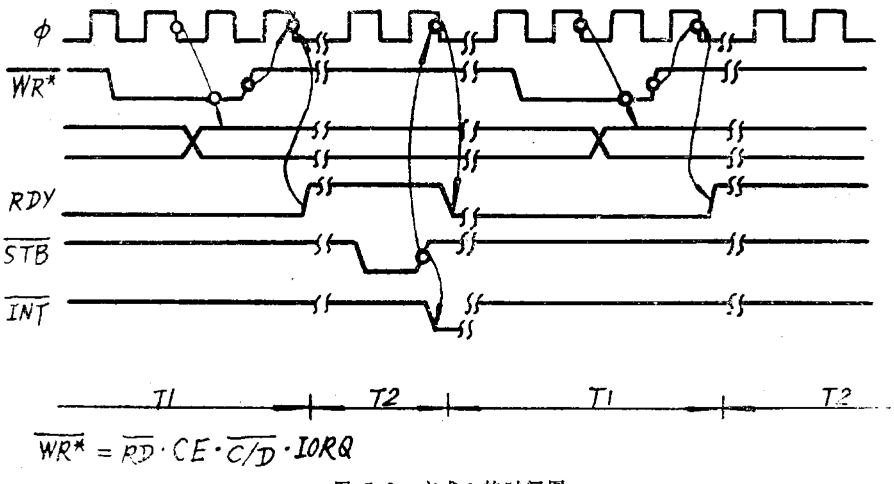


图 5.6 方式 0 的时间图

高，告知外部设备口 A 中已有数据可供使用。这一时间段如图 T1 中所示。此后 A-RDY 将保持有效，CPU 转入正常程序流程。当外部设备在 $\overline{ASTB}$ 线上发出一个负脉冲，将口 A 中数据取走后(即 $\overline{ASTB}$ 出现一个上升沿)，自动产生中断请求，即 $\overline{INT}$ 电平变低。这一时间段如图中 T2 所示。此后，如果 CPU 响应此中断请求，口 A 将中断矢量40与 CPU 的 I 寄存器中的 23 合成一个完整的中断矢量 2340H，在 2340 和 2341 单元中取出口A 中断服务程序的起始地址，例如，如表 5.4 所示为 2200。接着执行起始地址为 2200 的中断服务程序，CPU 进行适当的操作，并又向口 A 输出一个数据，此后的过程与上相同。

由上可见，外部设备与 CPU 之间的全部数据传送是在中断控制下实现的。尽管这种传送周期可以很长，但是它占用 CPU 的时间很少，因而很少影响 CPU 正常程序流程的进行。

二、方式 1—输入

以口 A 为例，图 5.7 示出了一个输入周期的时间图。这一周期是在 CPU 执行了一次读数之后，由外部设备利用 $\overline{A\ STB}$ 来启动。当 $\overline{A\ STB}$ 呈低电平时，从外部设备将数据装入口 A 内的数据输入寄存器，$\overline{A\ STB}$ 的上升沿将使 $\overline{INT}$ 电平变低。假如这时中断允许触发器已被置位，且这一器件的中断请求是优先权最高的，则下一 φ 的下降沿将使 A RDY 电平变低，表示数据输入寄存器已满，禁止再送数来。之后，CPU 在执行中

断服务程序过程中，从口 A 读取数据(发生在 $\overline{RD^*}$ 上出现一个负脉冲时)，在 $\overline{RD^*}$ 信号上升沿的下一个 ϕ 的下降沿时，使 ARDY 又变高，从而允许外部设备将新的数据装入口 A。

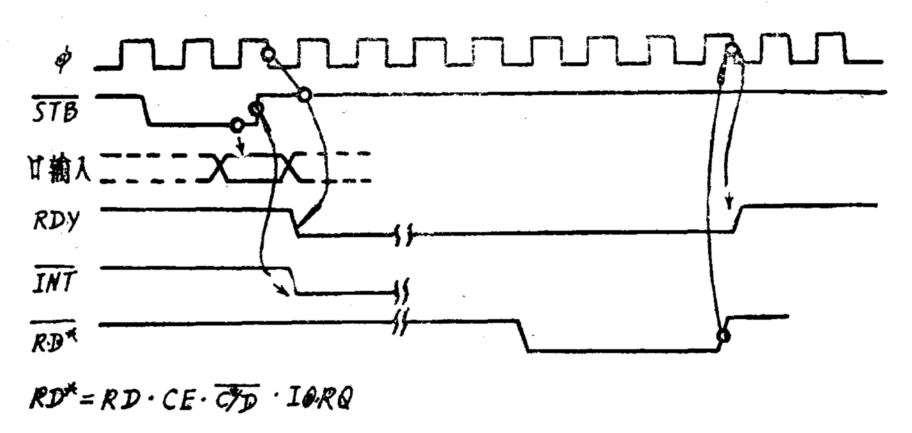


图 5.7 方式 1 的时间图

三、方式 2—双向操作

这一方式只有口 A 可使用。它只不过是方式 0 和方式 1 的组合而已，但必须使用四根联络线，因而除了使用口 A 的二根联络线 $\overline{\text{A STB}}$ 和 A RDY 作为输出的联络控制外，还需必占用口 B 的二根联络线 $\overline{\text{B STB}}$ 和 B RDY 作为口 A 输入的联络控制用。在这种情况下，口 B 只能工作在方式 3，因为方式 3 不需要使用联络线(见后)。

图 5.8 示出了这一方式的时间图。它与以上对方式 0 和方式 1 所介绍的情况几乎是相同的。其间的差别为：在方式 2 中，只有当 $\overline{\text{A STB}}$ 为低电平时，才允许数据送到口A的总线上。值得注意的是，必须将口 A 和口 B 的中断允许触发器置位(将开中断)，才能实现中断驱动的双向传送。

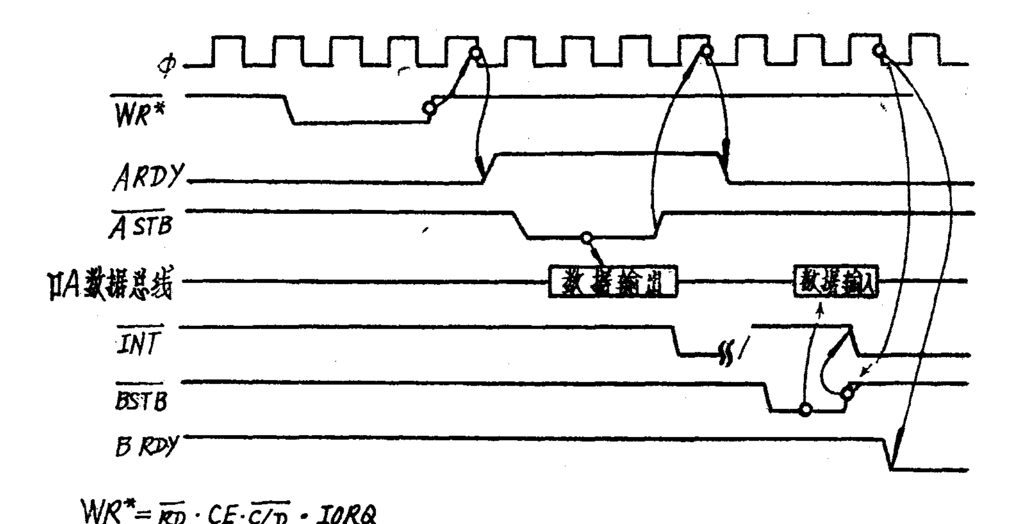


图 5.8 方式 2 的时间图

四、方式3—位控方式(以口B为例)

这一控制方式并不使用联络信号。在这方式中，可由程序规定口的某些线为输出线，另一些线为输入线。并可由程序规定，在外部设备中出现指定状态的情况时向CPU发出中断请求。

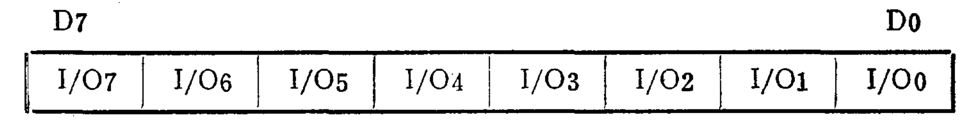
使用这一控制方式。可在任何时刻执行口子常规的写入或读出指令。在读PIO时，送回CPU的数据由两部分组成：一部分是数据输出寄存器中相应于被指定为输出位的内容；另一部分是数据输入寄存器中相应于被指定为输入位的内容。

现结合下面的初始化程序来说明PIO如何被设定成这种工作方式并进行工作，

```
LD      A，23H      设定中断矢量
LD      I，A
LD      A，CFH      设定方式3。下面送入的控制字不论其格式如何，必须
                    是一个I/O选择字
OUT     (83)，A
LD      A，29H      送入I/O选择字
OUT     (83)，A
LD      A，43H      设定中断矢量低字节
OUT     (83)，A
LD      A，B7H      设定中断控制字
OUT     (83)，A
LD      A，D6H      设定屏蔽字
OUT     (83，A
IM      2
EI
```

下面对只在方式3中使用的控制字进行解释：

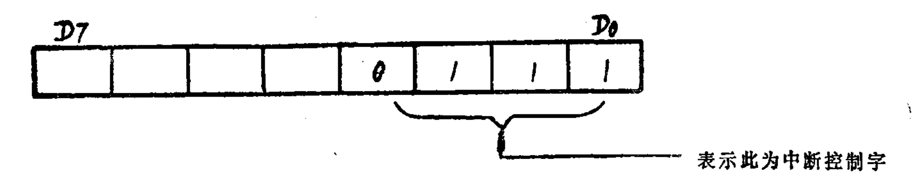
1．I/O选择字。此字紧跟在设定为工作方式3的控制字之后写入，它被锁存在前述的I/O选择寄存器中。



I/O=1 该位为输入

I/O=0 该位为输出

2．中断控制字。该字的D6D5锁存在屏蔽控制寄存器中。



D4＝1　　表示一个送入的控制字为中断屏蔽字

D4＝0　　其他情况

D5＝1　　被监视的信号高电平有效

D5＝0　　被监视的信号低电平有效

D6＝1　　对被监视的信号进行“与”操作

D6＝0　　对被监视的信号进行“或”操作

D7＝1　　开中断

D7＝0　　关中断

3．屏蔽字。此字紧跟在中断屏蔽字(且 D4＝1)之后写入，它被锁存在屏蔽寄存器中。

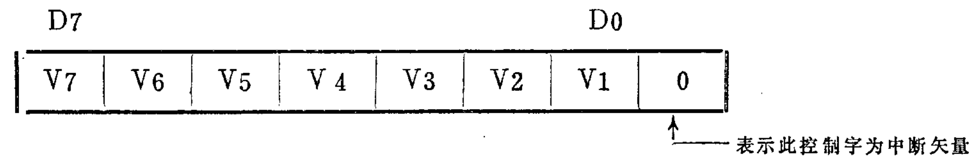
D7							D0
MB7	MB6	MB5	MB4	MB3	MB2	MB1	MB0

MB＝0　　该位受监视

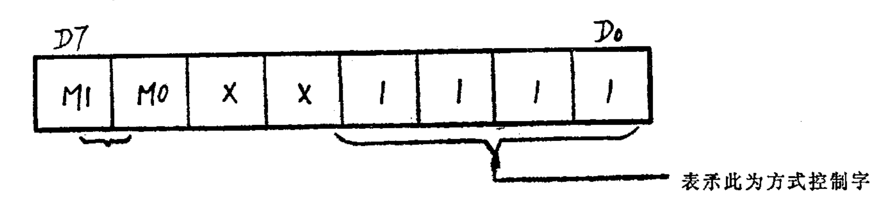
MB＝1　　该位的监视被屏蔽

五、程序设定 PIO 工作方式步骤

1．中断矢量



2．设置方式



00——输出

01——输入

10——双向

11——位控

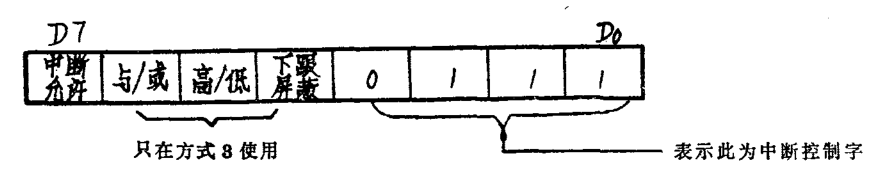
在选用方式 3(D7D6＝11)时下一控制字必须是 I/O 选择字：

D7							D0
I/O7	I/O6	I/O5	I/O4	I/O3	I/O2	I/O1	I/O0

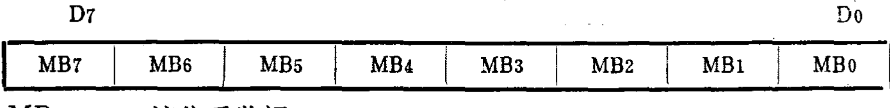
I/O＝1　　使该位为输入

I/O＝0　　使该位为输出

3．设置中断控制



若下跟屏蔽位(D4)为1，写入口的一个控制字必须是屏蔽字：

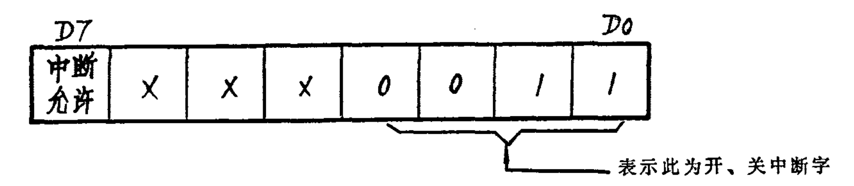


D7							D0
MB7	MB6	MB5	MB4	MB3	MB2	MB1	MB0

MB＝0　　该位受监视

MB＝1　　该位的监视被屏蔽

此外，口的中断允许触发器可以用下列命令来置1或置0，而不会修改中断控制字的其他部分。



5.4 PIO 的使用

本节以位控制方式的应用为例，说明 PIO 的使用及其电路连接。

图5.9示出了一个典型的控制方式的应用。假设要监视一个工业加工过程，任何不正常工作情况的发生，都将报告以 Z80-CPU 为中心所组成的控制系统。这过程的控制字

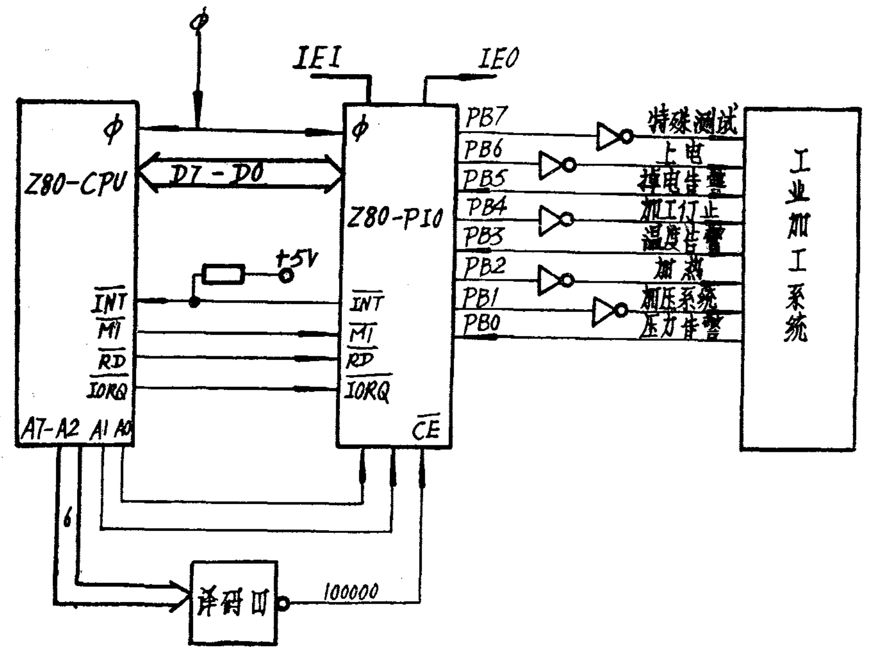


图 5.9　位控制方式的电路连接

具有下列内容：

1．方式控制字

1	1	×	×	1	1	1	1

选用方式 3，下一个必为 I/O 选择字；

2．I/O 选择字

0	0	1	0	1	0	0	1

PB0、PB3、PB5 为输入线。

3．中断矢量

0	1	0	0	0	0	1	0

4．中断控制字

1	0	1	1	0	1	1	1

表示对受监视的输入线进行“或”操作，并认为高电平有效。下面写入的控制字必须是一个屏蔽字。

5．屏蔽字

1	1	0	1	0	1	1	0

表示 PB0、PB3 和 PB5 线受监视。若其中任一根线为高电平都将产生中断请求。要求 CPU 进行告警处理。

此外，口 B 通过其输出线，向工业加工系统发出命令，指示它进行相应的操作。

第六章　计数器/定时器芯片 Z80-CTC

6.1 概　　述

在微型计算机化(Microcomputer—based)仪器和设备中，经常有一些定时和计数的要求，例如要求微型机驱动步进马达一类的电力机械，即要求微型计算机产生具有数毫秒脉宽的脉冲。微型计算机实现这类要求的具体方法有下列三种。

1. 利用等待循环(Wait loop)。这种方式的缺点是束缚了 CPU，使 CPU 在等待循环中不能为其他事件服务。

2. 利用单冲电路(One shot)。如图 6.1 所示，通过微型机的输出口产生正(或负)跳变，使单稳电路产生一脉冲。这种方法的缺点是脉冲宽度与单冲电路中的电路时间常数 RC(即图中的电阻与电容的乘积)有关，一经设定，不能由程序来更改。另外，单冲电路给微型计算机化设备的调试带来很大的麻烦。

3. 使用可编程序的计数/定时器。现代流行的微型计算机中都可包含有这样的芯片，它们可由程序来设定脉冲的个数、频率、波形等。

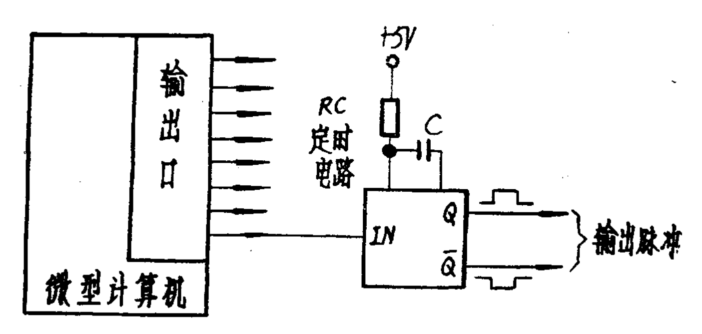


图 6.1　由单冲电听来产生脉冲

Z80—CTC 计数器/定时器(以下简称 CTC)是一种具有四个独立通道的可编程序器件。它有下列两个功能：

1. 定时功能。它能定时地发出脉冲(并能在发脉冲的同时请求中断)。脉冲的时间间隔、脉冲序列的起始和终了、产生脉冲的方式以及脉冲的个数均可由程序来设定。

2. 计数功能。它能对外界事件进行计数，当达到程序规定的数值时，向 CPU 请求中断(并可输出一脉冲)。

6.2 CTC 的方框图及引脚

CTC 的方框图示于图 6.2。它的每个通道都具有各自的中断矢量。0 号通道具有最

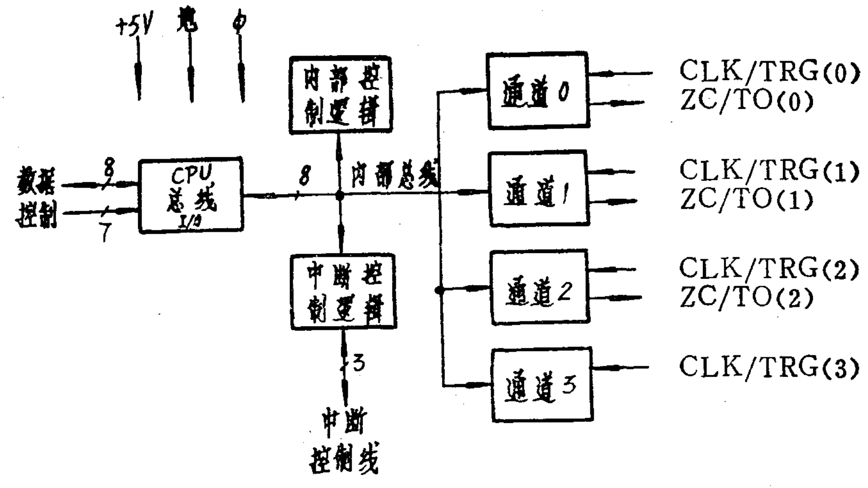


图 6.2 CTC 的框图

高的中断优先级。每个通道的方框图如图 6.3 所示。它由两个 8 位的寄存器、两个 8 位的计数器以及控制逻辑线路所组成。两个寄存器是时间常数寄存器和通道控制寄存器。两个计数器是减 1 计数器(CPU 可读)和定标器(仅用于定时器工作方式)。它们的功能将在下节操作说明中介绍。

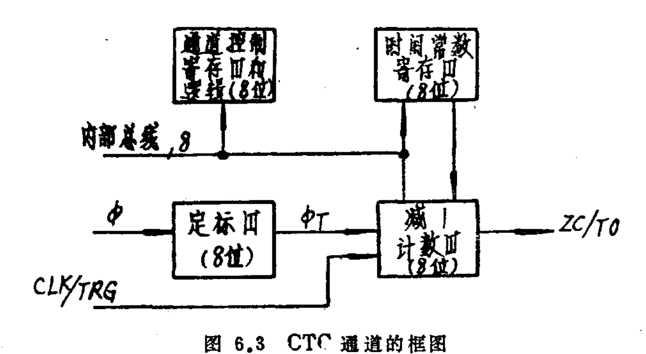


图 6.3 CTC 通道的框图

CTC 的引脚布置如图 6.4 所示。这些引脚的作用如下：

1. D7—D0 为 Z80—CPU 数据总线(双向，三态)。此总线用来在 CPU 和 CTC 之间传送数据和命令。

2. $\overline{CE}$ 为芯片允许(输入，低电平有效)。只有当此信号为低电平时，才允许 CPU 访问本 CPU 芯片。通常，这个信号是根据地址总线低 8 位中的 A7—A0 进行译码得出。在 CJ801 单板机系统中，当 A7—A2＝100001 时，才选中 CTC。

3. CS1，CS0 为通道选择(输入)。通常这两个引脚接至地址总线的 A1，A0 位，从而形成一个两位二进制地址码，以便在四个 CTC 独立通道中选择一个通道。其选择逻辑如表 6.1 所示。

4. $\overline{M1}$ 为 CPU 的取指令机器周期信号(输入，低电平有效)。

5. $\overline{IORQ}$ 为 CPU 的 I/O 请求信号(输入，低电平有效)。

6. $\overline{RD}$ 为 CPU 的读周期状态(输入，低电平有效)。

上述三种控制信号的作用如表 6.2 所示。

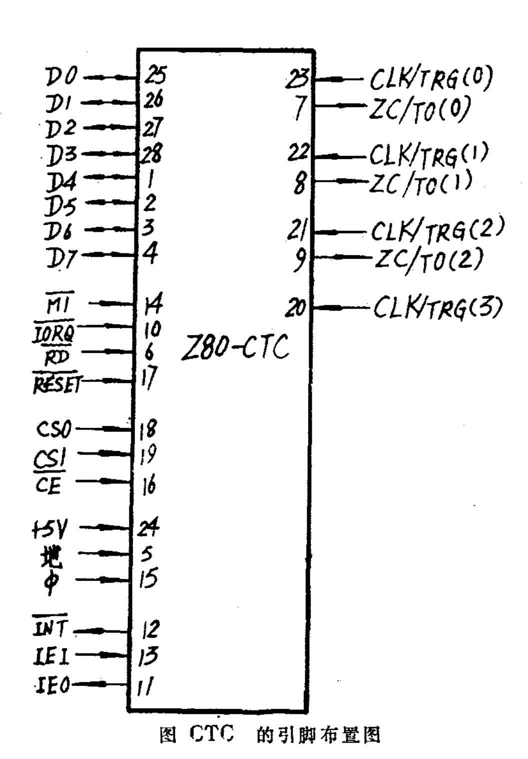


图 CTC 的引脚布置图

表 6.1　　　　CTC 的选择逻辑

引脚			选中单元	单元地址
$\overline{CE}$	CS1 (A 1)	CS0 (A 1)		
0(A7－A2＝100001)	0	0	通　道　0	84H
0(A7－A2＝100001)	0	1	通　道　1	85H
0(A7－A2＝100001)	1	0	通　道　2	86H
0(A7－A2＝100001)	1	1	通　道　3	87H
1(A7－A2＝100001)	×	×	芯片未被选中	

表 6.2　　　　CTC 的控制信号的作用

引脚			功　能　解　释
M1	IORQ	RD	
0	0	1	中断响应
0	1	0	检查中断服务程序是否结束
1	0	0	CPU 从 CTC 读出
1	0	1	从 CPU 写入 CTC
其他值			没有作用

7．IEI 为中断允许输入信号(输入，高电平有效)。

8．IEO 为中断允许输出信号(输出，高电平有效)。

9．$\overline{\text{INT}}$ 为中断请求(输出，漏极开路，低电平有效)。

上述三种控制信号的作用在第五章已经介绍，此处不再重述。顺便指出，通道0具有最高的中断优先级，依次为通道1，2，3。

10．$\overline{\text{RESET}}$为复位信号(输入，低电平有效)。这个信号终止所有通道的工作，禁止CTC产生中断请求，并禁止ZC/TO输出正脉冲。IEO重复IEI状态，同时CTC的数据总线输出驱动器变成高阻状态。

11．CLK/TRG(3－0)(或称C/T3－0)为四个通道的“外部时钟/定时器触发”脉冲(输入，可由用户规定其为正跳变或负跳变有效)。它们的作用将在下节介绍。

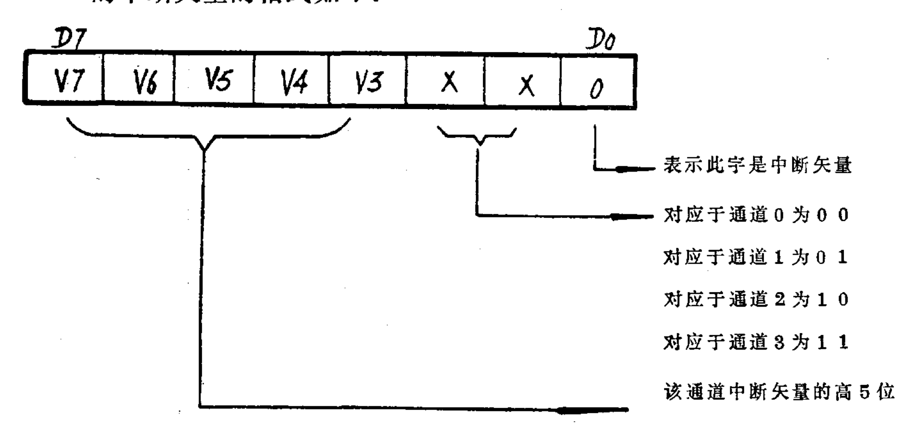
12．ZC/TO(2－0)(或称ZC2－0)为三个通道(通道3除外)的“回零/时间到”脉冲(输出)。它们的作用将在下节介绍。

13．其他：ϕ为时钟；＋5V为电源；GND为地。

6.3 CTC的操作说明

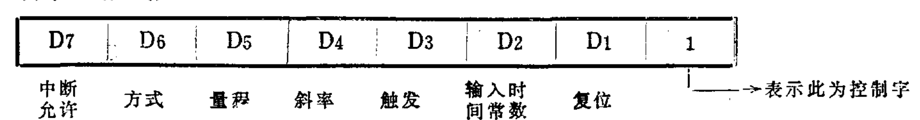
CTC的每个通道都有两种工作方式——定时器和计数器。从CPU向CTC写入的字有三种——控制字、时间常数和中断矢量每个通道只占用一个外部设备地址，而不象PIO的一个口占用两个外部设备地址——一个用来传送数据，另一个用来传送控制字。因而需要规定一些特征来区别上述三种字。

CTC的中断矢量的格式如下：



中断矢量的高五位在CTC程序设计时写入通道0(中断矢量只需、也只能写入通道0)，下二位将由中断控制逻辑提供对应于工作通道的二进制编码，如上述中断矢量格式所示。

控制字的格式如下：



D7＝1　允许通道中断

D7＝0　禁止通道中断

D6＝1　通道选择计数器方式工作

D6 = 0　　通道选择定时器方式工作

D5 = 1　　表示定标器系数为 256

D5 = 0　　表示定标器系数为 16

　　　　　(D5 仅在定时器方式时才有意义)

D4 = 1　　表示 CLK/TRG 的上升沿为有效

D4 = 0　　表示 CLK/TRG 的下降沿为有效

D3 = 1　　由 CLK/TRG 来启动定时器的工作

D3 = 0　　由装入时间常数来启动定时器的工作

　　　　　(D3 仅在定时器方式时才有意义)

D2 = 1　　下一个写入的字是时间常数(或初始常数)

D2 = 0　　下一个写入的字不是时间常数

D1 = 1　　立即停止通道的工作，ZC/TO 不起作用，并禁止通道中断逻辑。

D1 = 0　　每当减 1 计数器变为零时，时间常数寄存器立即将其内容装入减 1 计数器，通道继续现行操作。

可见时间常数紧跟在控制字(且 D2 = 1)之后写入，而不须用特征来表示。

下面介绍 CTC 的两操作方式。

一、定时器工作方式

例 1．由程序装入时间常数来启动定时器工作(以通道 0 为例)。

程序设计

```
LD    A, 23H     设定中断矢量
LD    I, A
LD    A, 50H
OUT   (84), A
LD    A, 85H     设定工作方式
OUT   (84), A
LD    A, 03H     装入时间常数
OUT   (84), A
IM    2
EI
```

定时器工作的时间图如图 6.5 所示。

定时器的工作过程是：当时间常数 03_H 装入时间常数寄存器后，定时器开始工作，时间常数寄存器将其内容装入减 1 计数器，定标器对时钟 ϕ 进行计数，本例中每输入 16 个 ϕ(因为控制字 D5 = 0)，定标器输出一个 ϕ_T(见图 6.3)使减 1 计数器减 1，定标器输出第三个 ϕ_T 时，减 1 计数器由1减为 0，ZC/TO 发出一正脉冲，$\overline{\text{INT}}$ 电平变低，向 CPU 请求中断。本例中因为控制字 D1 = 0，时间常数寄存器将其内容再一次装入减 1 计数器，并重复上述操作。可见 ZC/TO 上每隔 48 个 ϕ 出现一个正脉冲，直到写入一个停止其操作的控制字(D1 = 1)为止。

本例中 ZC/TO 上发出正脉冲的时间间隔可以由程序来设定。发脉冲的起始和终了也

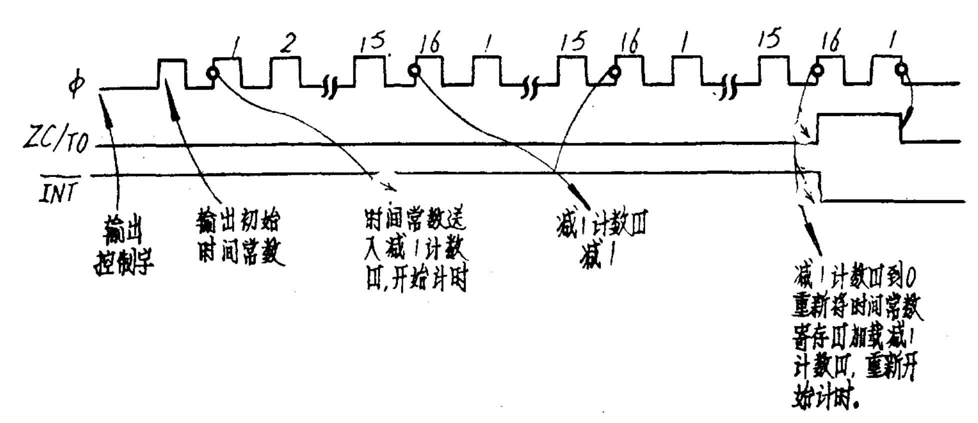


图 6.5 由程序控制的定时器的时间图

由程序决定。

例 2. 用外界 CLK/TRG 来启动定时器的工作(以通道 0 为例)。

程 序 设 计

```
LD    A, 23H     设计中断矢量
LD    I, A
LD    A, 40H
OUT   (84), A
LD    A, BDH     设计工作方式
OUT   (84), A
LD    A, 03H     装入时间常数
OUT   (84), A
IM    2
IE
```

定时器工作的时间图如图 6.6 所示。

定时器的工作过程如下：当时间常数 03H 装入时间常数寄存器后，定时器等待 LK/TRG 信号的到来。一旦出现 CLK/TRG 信号(本例中控制字 D4＝1，故正跳变有

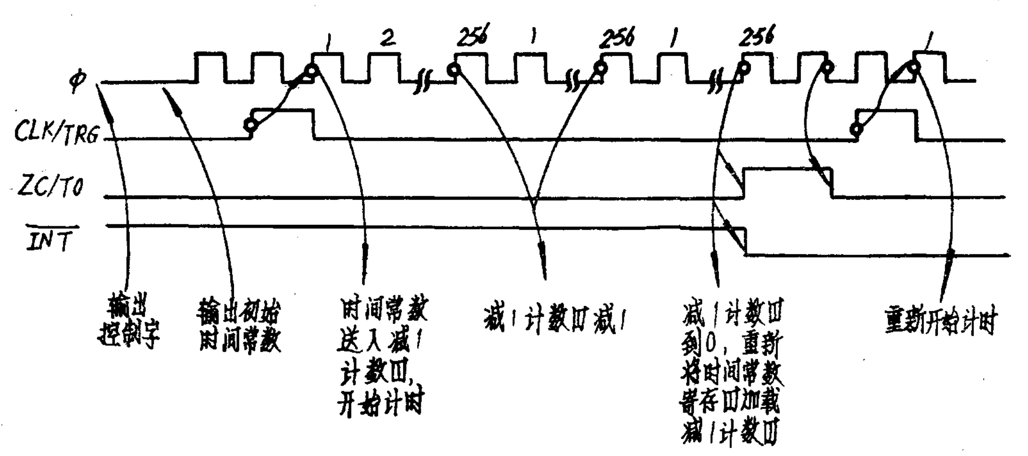


图 6.6 CLK/TRG 由外界控制的定时器的时间图

效)，定时器开始工作，其后的过程与例1基本相同。所不同的是，其一，每当出现第256个 φ 后(而不是第16个，因为控制字中 D5=1)减1计数器减1；其二，当减1计数器为零后，ZC/TO 上出现正脉冲，$\overline{\text{INT}}$ 电平变低，定时器停止工作。只有当再一次出现 CLK/TRG 信号时，才能再次启动定时器的工作。

二、计数器工作方式

这种工作方式主要用于对外界(异步)事件进行计数。当到达规定数值后，ZC/TO 发出正脉冲，并使 $\overline{\text{INT}}$ 电平变低。这种工作方式的时间图如图 6.7 所示。

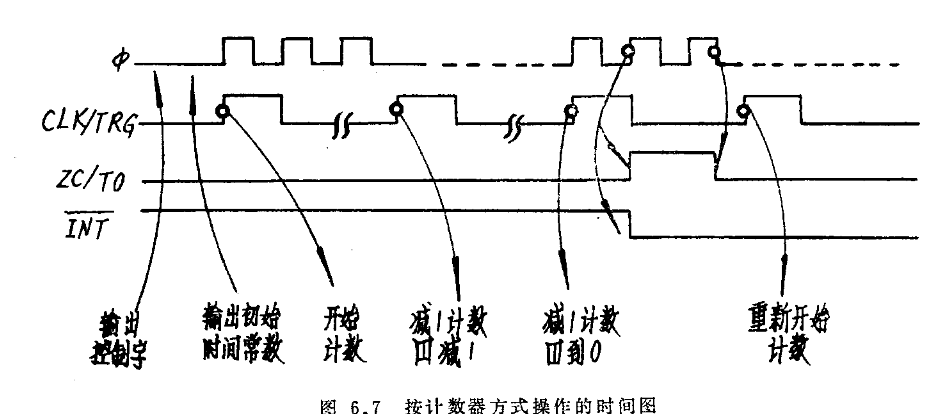


图 6.7 按计数器方式操作的时间图

程 序 设 计

```
LD    A, 23H     设定中断矢量
LD    I, A
LD    A, 40H
OUT   (84), A
LD    A, D5H     设定工作方式
OUT   (84), A
LD    A, 03H     装入初始常数
OUT   (84), A
IM    2
EI
```

计数器的工作过程是：首先输入控制字，规定按计数器方式工作等等。其次输入初始常数，计数器开始工作，时间常数寄存器将其中的初始常数装入减1计数器。之后每当输入一个 CLK/TRG 信号(表示发生一次外界事件)，减1计数器即减1。直到减1计数器到达零时，ZC/TO 发出一正脉冲，$\overline{\text{INT}}$ 电平变低，时间常数寄存器再次将时间常数装入减1计数器，重复上述数操作。

6.4 CTC 的硬件连接

图6.8示出了 CPU 与 CTC 的硬件连接。它和 PIO 的连接图相近，此处不再赘述。

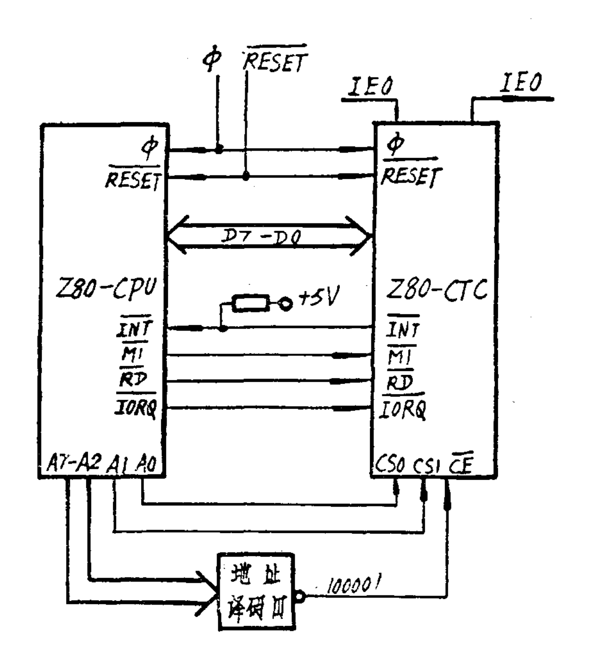


图 6.8 CTC 与 CPU 的硬件连接图

点阵式打印机在微型机系统中的应用

电工及计算机科学系　张华宋　吴报鑫
长江电子计算机厂　赵金荣

μ80，EG602等类型点阵式打印机中配有微处理器，藉以控制点阵式打印机芯，实现传统打印机由机械方式完成的任务。它们具有体积小，功能强、噪音低等优点，适用于台式小型计算机系统。

一、打印机的主要性能

机型 性能	μ 80	EG602
1．字符尺寸：	9×7点阵	9×7点阵
2．打印速度：	28行/分（每行80个字符） 51行/分（每行40个字符）	30个字符/秒
3．行距：	6行/吋及8行/吋	6行/吋（字符码） 9行/吋（作图码）
4．字体：	扩展字体——40字符/行 正常字体——80字符/行 压缩字体——132字符/行	扩展字体—40字符/行 正常字体—80字符/行
5．打印符号：	52个英文大、小写字母 44个阿拉伯数字及其它符号 } 见表一 64个日本片段名字符和符号及64个示形符号(1)	英文的小写字母，阿拉伯数字及其它符号 美、英、德、瑞典文特殊符号及作图符号(2)
6．外形尺寸：	342×245×108mm³	172.5×328×132mm³
7．重量：	6.5kg	2.5kg

表一、μ 80 部分字符集

	b8 b7 b6 b5	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1
b4b3b2b1	R\C	0	1	2	3	4	5	6	7
0 0 0 0	0			空	0	@	P		p
0 0 0 1	1			!	1	A	Q	a	q
0 0 1 0	2			"	2	B	R	b	r
0 0 1 1	3			#	3	C	S	c	s
0 1 0 0	4			$	4	D	T	d	t
0 1 0 1	5			%	5	E	U	e	u
0 1 1 0	6			&	6	F	V	f	v
0 1 1 1	7			'	7	G	W	g	w
1 0 0 0	8			)	8	H	X	h	x
1 0 0 1	9			(	9	I	Y	i	y
1 0 1 0	A	LF		*	:	J	Z	j	z
1 0 1 1	B		ESC	+	;	K	↑	k	{
1 1 0 0	C			,	<	L	↓	l	I
1 1 0 1	D	CR	GS	—	=	M	←	m	}
1 1 1 0	E		RS	.	<	N	↓	n	—
1 1 1 1	F		US	/	?	O	—	o	空

特殊符号注释

符号	R\C	说　　明
"	22	引号
'	27	撇号
,	2C	逗号
—	2D	负号，划线
.	2E	句点
—	5F	下划
`	60	重音符
—	7E	上划
LF	OA	换行
CR	OD	回车
GS	1E	压缩字体
RS	1D	正常字体
US	1F	扩展字体
ESC 6	1B 36	6行/吋
ESC 8	1B 38	8行/吋
ESC A	1B 41	长行(80字符/行)
ESC B	1B 42	短行(对称 60 字符/行)

EG602 的打印字符与 μ 80 基本相似，所不同的　具有德文、瑞典文中的特殊字符，并能根据点阵位置打印汉字的各种图形。详见参考资料(2)。

二、Z—80 单板机与打印机的接口

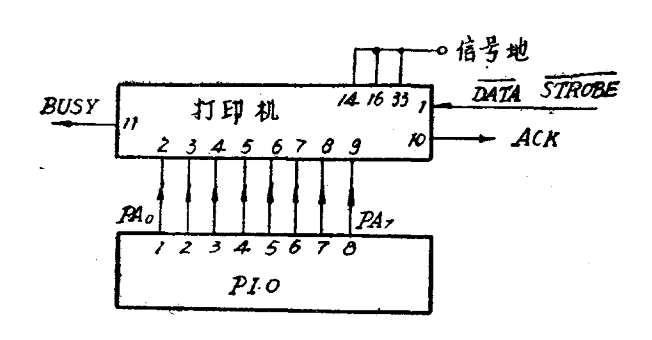
主机与打印机之间的数据传送有串行和并行两种。为提高传送速度，通常采用并行传送方式。

Z—80 单板机与打印机的连接可采用 PIO 或 8212 作为接口器件。现分别介绍如下：

1. PIO 作为接口器件

PIO 有二个口(口A、口B)和四种操作方式(输入、输出、双向、位控)。在用作打印机接口器件时，应采用输出方式(方式0)；而口 A 或口 B 任选。

μ80 或 EG602 打印机有 36 个接线端子，通常只需连接 14 根线(见图一)，就可得到良好的打印效果。



图一 PIO作为打印机连接口器件

8 根数数据线 D_0～D_7(2 脚～9 脚)。用于接收由 Z—80CPU 发出，经 PIO 传送的信息。

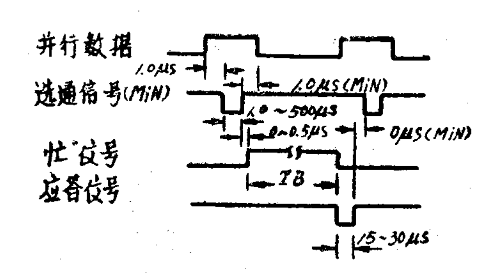
1 根数据选通信号线 Data Strobe(1 脚)。CPU 在发送数据的同时，还必需通过选通线向打印机发出选通脉冲，通知它接收数据。

1 根“忙”信号线 BUSY(11 脚)。当打印机正在执行某种功能而不能接收据数时，BUSY 端呈高电平。在采用查询方式风印时，可通过程序来查询 BUSY 端的状态。而采用等待方式打印时，则可将 BUSY 信号进行电平转换后接 CPU 的 WAIT 端。

1 根应答信号线 ACK(10 脚)。打印机接收数据并存入缓冲存贮器后(或执行打印命令后)都发出 ACK 应答信号，表示数据已被接收(或打印已被执行)。在采用中断方式打印时，需利用 ACK 端提供中端申请信号。

3 根信号地线(14，16，33 脚)。接信号低电平。

图二画出了数据信号、选通信号。“忙”信号、应答信号之间的时序图。在进行调试时，只要能满足这种时序关系，就能得到良好的打印效果。

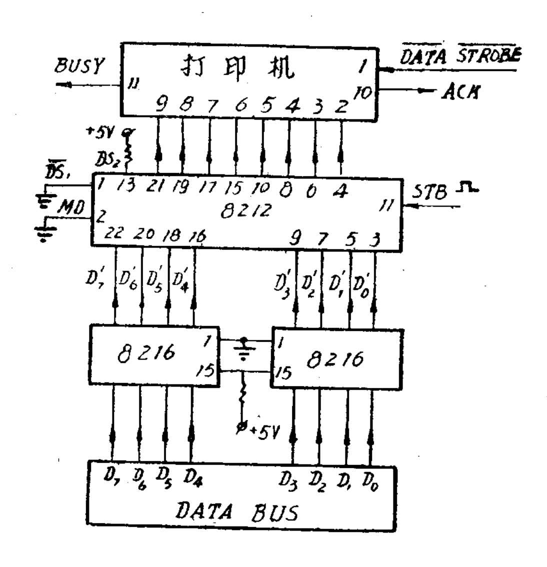


图二 信号时序图

2．8212 作为接口器件(图三)

8212 八位输入/输出器能用来把数据总线上的数据发送供外部设备——打印机。8212

作为输出口时，器件选择 DS_1 端置低电位，DS_2 置高电位，方式 MD 置低电位。当单板机发送一个选通脉冲给 STB 脚时，8212 的输出数据就等于其输入端的数据(见图三)。图中二块 8216 固件是用来缓冲微型机系统器件的四位双向母线驱动器。



图三　8212作为接口器件

三、软件设计

当单板机与打印机之间通过硬件接口联机后，把相应程序输入单板机。在执行指令的过程中、单板机将控制打印机有条不紊地完成打印功能。

软件设计的基本要点如下：

1．设置 PIO 口的工作方式(若用 8212 作为接口器件，则不必通过程序预置工作方式，但需注意在硬件连接时将 8212 按输出方式接线)。

2．如果打印数据是二进制码，需调用相应子程序将二进制数转换为 ASCII 码(CALL UBASC)。

3．调用送数子程序(CALL DS)。

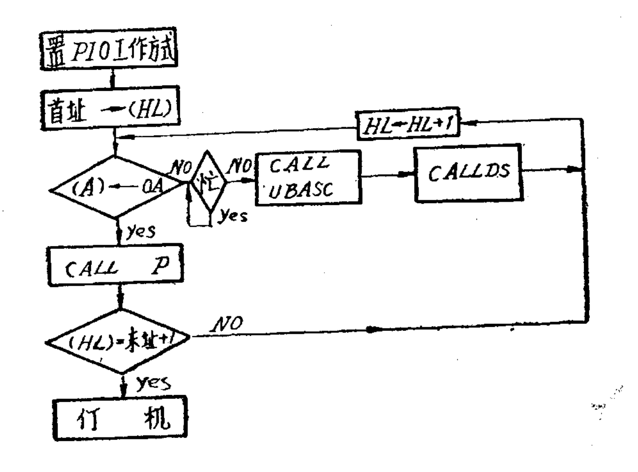
4．CPU 一旦收到打印字符(0A)，单板机立即调用打印子程序(CALL P)，于是印机将缓冲存贮器里的字符打印出来并执行。

5．重复步骤 2，直至指定存贮单元的内容全部打印完毕。

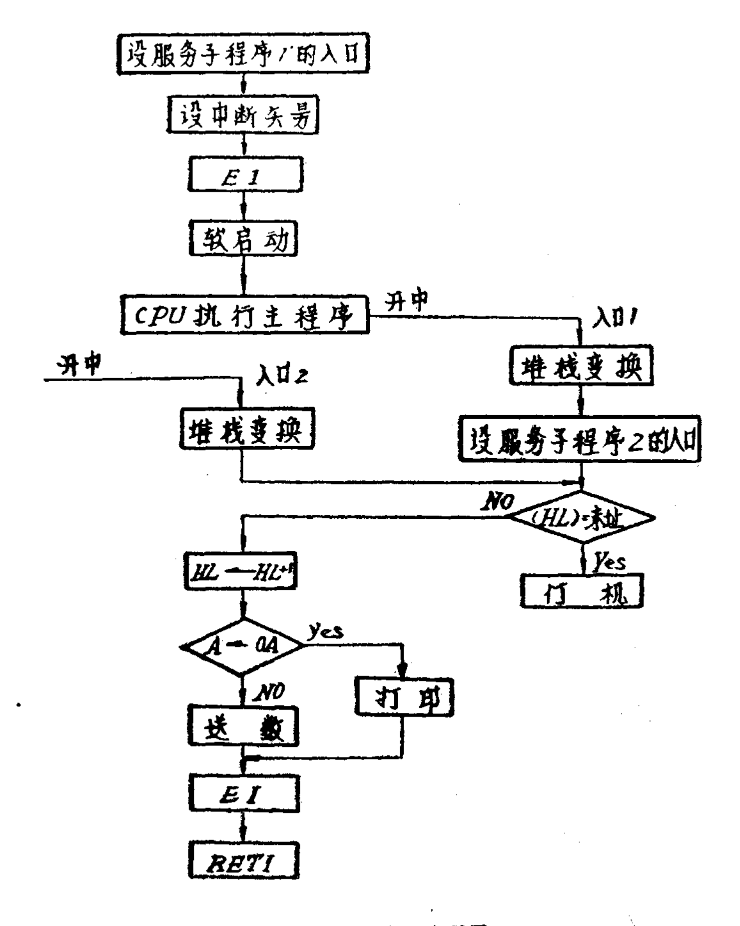
图四画出了查询方式打印管理程序的简略流程图

在指定数据存贮单元中，可按所要求打印的字体、行距、每行字符数等设置数据格式(见表二)，然后再放被打印的字符。若要改变打印格式，需在一行数据的前面重新设置相应数据格式。

图五为中断方式打印管理程序的流程图



图四 打印管理程序流程图(查程方式)



图五 中断方式流程图

四、小　　结

点阵式打印机与 Z—80 单板机联机后，一般有三种工作方式：等待、查询和中断方式。其中等待方式又分软件等待和硬件等待两种。所谓软件等待是在送数子程序（DS）和打印子程序(P)中，设置一定的时间延迟。这种方式的打印速度慢，机动性差，但接线简单。硬件等待、查询和中断这三种方式的打印速度相近，其中以硬件等待方式的程序最简单，当然其硬件连线略复杂些。中断方式的打印管理程序较复杂、但它的最大优点是合理解决快速运算的 CPU 与低速工作的打印机等外设之间的矛盾，可以使多个外围设备与主机并行工作，大大提高了 CPU 的使用效率。因此在要求并行工作，提高效率的场合，一般应用中断制方式。

参　考　文　献

(1) μ—80 MICROLINE PRINTFRO WNER'S MANUAL

(2) OWNER'S MANUAL EG602 GRAPHIC PRINTER

CJ801

Z80 单板计算机设计手册

Z80 微型计算机袖珍使用手册

主寄存器组		辅助寄存器组		
累加器 A	标 志 F	累加器 A′	标 志 F′	
B	C	B′	C′	通 用 寄存器
D	E	D′	E′	
H	L	H′	L′	

中 断 矢 量 I	存储器刷新 R	
变址寄存器 IX		专 用 寄存器
变址寄存器 IY		
栈 指 示 器 SP		
程序计数器 PC		

z80-CPU 寄存器的配置

标志操作摘要

指　　令	S (D7)	Z		H		P/V	N	C (D0)	说　　明
ADD A, s; ADCA, s	↕	↕	X	↕	X	V	0	↕	8位加或带进位加
SUBs;SBCA,s;CPs;NEG	↕	↕	X	↕	X	V	1	↕	8位减，带进位减，比较和累加器求补
ANDs	↕	↕	X	1	X	P	0	0	}逻辑运算
ORs; XORs;	↕	↕	X	0	X	P	0	0	
INCs	↕	↕	X	↕	X	V	0	•	8位增量
DECs	↕	↕	X	↕	X	V	1	•	8位减量
ADD HL, SS	•	•	X	X	X	•	0	↕	16位加
ADC HL, SS	↕	↕	X	X	X	V	0	↕	16位带进位加
SBC HL, SS	↕	↕	X	X	X	V	1	↕	16位带进位减
RLA;RLCA;RRA;RRCA	•	•	X	0	X	•	0	↕	累加器循环移位
RLs;RLCs;RRs;RRCs; SLAs; SRAs; SRLs	↕	↕	X	0	X	P	0	↕	存储单位循环和移位
RLC; RRD	↕	↕	X	0	X	P	0	•	向左和向右循环移位数字
DAA	↕	↕	X	↕	X	P	•	↕	十进制调整累加器
CPL	•	•	X	1	X	•	1	•	累加器变反
SCF	•	•	X	0	X	•	0	1	进位位置位
CCF	•	•	X	X	X	•	0	↕	进位位变反
INr, (C)	↕	↕	X	0	X	P	0	•	输入寄存器接间寻址
INI; IND; OUTI; OUTD	X	↕	X	X	X	X	1	•	}数据块输入和输出
INIR;INDR;OTIR;OTDR	X	1	X	X	X	X	1	•	}如B≠0，则Z＝0；否则Z＝1
LDI; LDD	X	X	X	0	X	↕	0	•	}数据块转移指令
LDIR; LDDR	X	X	X	0	X	0	0	•	}如BC≠0，则P/V＝1；否则P/V＝0
CPI;CPIR;CPD;CPDR	X	↕	X	X	X	↕	1	•	数据块查找指令如A＝(HL)，则Z＝1；否则Z＝0。如BC≠0，则P/V＝1；否则P/V＝0
LDA, I; LDA, R	↕	↕	X	0	X	IFF	0	•	中断允许触发器(IFF)的内容复制到P/V标志
BITb, s	X	↕	X	1	X	X	0	•	存储单元S的b位状态复制到Z标志

上页表所用的符号如下：

符号	操　作
C	进位/借位标志，如果运算中的操作数或结果的最高有效位产生进位，则C＝1。
Z	零标志。如果运算结果为零，则Z＝1
S	符号标志。如果运算结果的最高有效位为1，则S＝1。
P/V	奇偶或溢出标志。奇偶(P)和溢出(V)共用此标志。逻辑运算将视运算结果的奇偶性来改变其状态，而算术运算则将视结果的溢出与否改变其状态。如果P/V表示奇偶性，则当运算结果为偶数时P/V＝1，奇偶时P/V＝0；如P/V表示溢出状态，则当运算产生溢出时P/V＝1。
H	半进位标志。如果加或减运算结果向累加器的第4位进位或从该位借位，则H＝1。
N	加/减标志。如果前一运算是减，则N＝1。 H和N标志和十进制调整指令(D A A)联用，将BCD格式的操作数的加或减的结果修正为BCD结果。
↕	根据运算结果来影响标志状态。
•	运算结果不影响标志。
0	运算使标志复位。
1	运算使标志置位。
X	此标志可处于随意状态。
V	根据运算溢出与否影响P/V标志。
P	根据运算结果的奇偶影响P/V标志。
r	CPU寄存器A、B、C、D、E、H和L中的任何一个。
s	以特定指令所允许采用的各种寻址方式寻址的任何8位存储单元。
ss	以特定指令所允许采用的各种寻址方式的任何16位存储单元。
ii	两个变址寄存器IX和IY中的任何一个。
R	刷新计数器
n	(0，255)范围内的8位数值。
nn	(0，65535)范围内的16位数值。

8位传送指令组(LD)

源

目的地		隐含		寄存器							寄存器间接			变址		扩展寻址	立即
		I	R	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX+d)	(IY+d)	(nn)	n
寄存器	A	ED 57	ED 5F	7F	78	79	7A	7B	7C	7D	7E	0A	1A	DD 7E d	FD 7E d	3A n n	3E n
	B			47	40	41	42	43	44	45	46			DD 46 d	FD 46 d		06 n
	C			4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d		0E n
	D			57	50	51	52	53	54	55	56			DD 56 d	FD 56 d		16 n
	E			5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d		1E n
	H			67	60	61	62	63	64	65	66			DD 66 d	FD 66 d		26 n
	L			6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d		2E n
寄存器间接	(HL)			77	70	71	72	73	74	75							36 n
	(BC)			02													
	(DE)			12													
变址	(IY+d)			DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d							DD 36 d n
	(IY+d)			FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d							FD 36 d n
扩展寻址	(nn)			32 n n													
隐含	I			ED 47													
	R			ED 4F													

8位传送指令组

助记符	用符号表示的操作	标志								操作数				字节数	M周期数	T状态码	说明
		S	Z		H		P/V	N	C	76	543	210	Hex				
LDr, s	r←s	•	•	X	•	X	•	•	•	01	r	s		1	1	4	r,s寄存器
LDr, n	r←s	•	•	X	•	X	•	•	•	00	r	110		2	2	7	000B
										←	n	→					0001C
LDr, (HL)	r←(HL)	•	•	X	•	X	•	•	•	01	r	110		1	2	7	010D
LDr,,(IX+d)	r←(IX+d)	•	•	X	•	X	•	•	•	11	111	101	DD	3	5	19	011E
										01	r	110					100H
										←	d	→					101L
LDr,(IY+d)	r←(IY+d)	•	•	X	•	X	•	•	•	11	011	101	FD	3	5	19	111A
										01	r	110					
										←	d	→					
LD(HL), r	(HL)←r	•	•	X	•	X	•	•	•	01	110	r	DD	1	2	7	
LD(IX+d,)r	(IX+d)←r	•	•	X	•	X	•	•	•	11	011	101		3	5	19	
										01	110	r					
										←	d	→	FD				
LD(IY+d),r	(IY+d)←r	•	•	X	•	X	•	•	•	11	111	101		3	5	19	
										01	110	r					
										←	d	→	36				
LD(HL), n	(HL)←n	•	•	X	•	X	•	•	•	00	110	110		2	3	10	
										←	n	→					
LD(IX+d), n	(IX+d)←n	•	•	X	•	X	•	•	•	11	011	101	DD	4	5	19	
										00	110	110	36				
										←	d	→					
										←	n	→					
LD(IY+d), n	(IY+d)←n	•	•	X	•	X	•	•	•	11	111	101	FD	4	5	19	
										00	110	110	36				
										←	d	→					
										←	n	→					
LDA, (BC)	A←(BC)	•	•	X	•	X	•	•	•	00	001	010	0A	1	2	7	
LDA, (DE)	A←(DE)	•	•	X	•	X	•	•	•	00	011	010	1A	1	2	7	
LDA(nn)	A←(nn)	•	•	X	•	X	•	•	•	00	111	010	3A	3	4	13	
										←	n	→					
										←	n	→					
LD(BC),A	(BC)←A	•	•	X	•	X	•	•	•	00	000	010	02	1	2	7	
LD(DE),A	(DE)←A	•	•	X	•	X	•	•	•	00	010	010	12	1	2	7	
LD(nn),A	(nn)←A	•	•	X	•	X	•	•	•	00	110	010	32	3	4	13	
										←	n	→					
										←	n	→					
LDA, I	A←I	↕	•	X	0	X	IFF	0	•	11	101	101	ED	2	2	9	
										01	010	111	57				
LDA, R	A←R	↕	↕	X	0	X	IFF	0	•	11	101	101	ED	2	2	9	
										01	011	111	5F				
LDI, A	I←A	•	•	X	•	X	•	•	•	11	101	101	ED	2	2	9	
										10	000	111	47				
LDR, A	R←A	•	•	X	•	X	•	•	•	11	101	101	ED	2	2	9	
										01	001	111	4F				

注：r，s表示A、B、C、D、E、H、L寄存器中的任何一个，IFF表示中断允许触发器的内容复制到P/V标制内。标志记号：• =标志不受影响，0=标志复位，1=标志置位，X=标复状态随意，↕=根据操作结果改变标志。

16 位传送指令组(LD, PUSH, POP)

源

目的地		寄	存	器					立即 扩充	扩展 地址	寄存器 间　接
		AF	BC	DE	HL	SP	IX	IY	nn	(nn)	(SP)
寄存器	AF										F1
	BC								01 n n	ED 4B n n	C1
	DE								11 n n	ED 5B n n	D1
	HL								21 n n	2A n n	E1
	SP				F9		DD F9	FD F9	31 n n	ED 7B n n	
	IX								DD 21 n n	DD 2A n n	DD E1
	IY								FD 21 n n	FD 2A n n	FD E1
扩展 地址	(nn)		ED 43 n n	ED 53 n n	22 n n	ED 73 n n	DD 22 n n	FD 22 n n			
压入指令→ 寄存器 间　接	(SP)	F5	C5	D5	E5		DD E5	FD E5			

↑
弹出指令

注：每次执行之后，压入与弹出指令调整 SP(栈指示器)

16位 传 传 指 令 组

助 记 符	用符号表示的操作	标志 S	Z		H		F/V	N	C	操作码 76 543 210	Hez	字节数	周期数	T状态数	说 明
LDdd, nn	dd←nn	•	•	X	•	X	•	•	•	00 dd0 001 ← n → ← n →		3	3	10	dd 寄存器对 00 BC 01 DE 10 HL 11 SP
LDIX, nn	IX←nn	•	•	X	•	X	•	•	•	11 011 101 00 100 001 ← n → ← n →	DD 21	4	4	14	
LDIY, nn	IY←nn	•	•	X	•	X	•	•	•	11 111 101 00 100 001 ← n → ← n →	FD 21	4	4	14	
LDHL,(nn)	H←(nn+1) L←(nn)	•	•	X	•	X	•	•	•	00 101 010 ← n → ← n →	1A	3	5	16	
LDdd,(nn)	dd_H←(nn+1) dd_L←(nn)	•	•	X	•	X	•	•	•	11 101 101 01 dd1 011 ← n → ← n →	ED	4	6	20	
LDIX, (nn)	IX_H←(nn+1)	•	•	X	•	X	•	•	•	11 011 101 00 101 010 ← n → ← n →	DD 2A	4	6	20	
LDIY, (nn)	IY_H←(nn+1)I IY_L←(nn)	•	•	X	•	X	•	•	•	11 111 101 00 101 010 ← n → ← n →	FD 2A	4	6	20	
LD(nn),HL	(nn+1)←H (nn)←L	•	•	X	•	X	•	•	•	00 100 010 ← n → ← n →	22	3	5	16	
LD(nn),dd	(nn+1)←dd_H	•	•	X	•	X	•	•	•	11 101 101	ED	4	6	20	

续 表

助记符	用符号表示的操作	S	Z		H		P/V	N	C	操作码 76 543 210	Hex	字节数	周期数	T状态数	说明
	(nn)←dd_L									01 dd0 011 ← n → ← n →					
LD(nn),IX	(nn+1←IX_H (nn)←IX_L	•	•	X	•	X	•	•	•	11 011 101 00 100 010 ← n → ← n →	DD 22	4	6	20	
LD(nn),IY	(nn+1)←IY_H (nn)←IY_L	•	•	X	•	X	•	•	•	11 111 101 00 100 010 ← n → ← n →	FD 22	4	6	20	
LDSP, HL	SP←HL	•	•	X	•	X	•	•	•	11 111 001	F9	1	1	6	
LDSP, IX	SP←IX	•	•	X	•	X	•	•	•	11 011 101 11 111 001	DD F9	2	2	10	
LDSP, IY	SP←IY	•	•	X •	•	X	•	•	•	11 111 101 11 111 001	FD F9	2	2	10	
PUSHqq	(SP−2)←qqL (SP−1)←qqH	•	•	X X	•	X	•	•	•	11 qq1 101		1	3	11	qq 寄存器对 00 BC 01 DE 10 HL 11 AF
PUSHIX	(SP−2)←IXL (SP−1)←IXH	•	•	X	•	X	•	•	•	11 011 101 11 100 101	DD E5	2	4	15	
PTSHIY	(SP−2)←IYL (SP−1)←IYH	•	•	X	•	X	•	•	•	11 111 101 11 100 101	FD E5	2	4	15	
POPqq	ppH←(SP+1) qqL←(SP)	•	•	X	•	X	•	•	•	11 qq0 001		1	3	10	
POPIX	IX_H←(SP+1) IX_L←(SP)	•	•	X	•	X	•	•	•	11 011 001 11 100 001	DD E1	2	4	14	
POPIY	IY_H←(SP+1) IY_L→(SP)	•	•	X	•	X	•	•	•	11 111 101 11 100 001	FD E1	2	4	14	

注：dd 是 BC、DE、HL和 ST 寄存器对中任一对，qq 是 AF、BC、DE 和 HL，寄存器对中的任一对，$(PAIR)_H$，$(PAIR)_L$ 分别表示寄存器对的高八位和低八位。如 $BC_L = C$，$AF_H = A$

标志记号同前

交换指令组(EX 和 EXX)

		隐含寻址				
		AF′	BC′，DE′ &HL′	HL	IX	IY
隐含	AF	D8				
隐含	BC DE & HL		D9			
隐含	DE			EB		
寄存器间接	(SP)			E3	DD E3	FD E3

数据块传送指令组

源

			寄存器间接 (HL)	
目的地	寄存器间接	(DE)	ED A0	传送 ‘LDI’-(DE)⟵(HL) HL 和 DE 增 1，BC 减 1
目的地	寄存器间接	(DE)	ED B0	传送 ‘LDIR’-(DE⟵(HL) HL 和 DE 增 1，BC 减 1， 如此重复到 BC = 0
目的地	寄存器间接	(DE)	ED A8	传送 ‘LDD’-(DE)⟵(HL) HL 和 DE 减 1，BC 减 1
目的地	寄存器间接	(DE)	ED B8	传送 ‘LDDR’-(DE)⟵(HL) HL 和 DE 减 1，BC 减 1 如此重复到 BC = 0

HL　指示源地址
DE　指示目的地址
BC　字节计数器

数据块搜索指令组

查找单元

寄存器间接 (HL)	
ED A1	‘CPI’ HL 增 1，BC 减 1
ED B1	‘CPIR’-HL 增 1， BC 减 1，重复直到 BC = 0 或找到需要的字符
ED A9	‘CPD’ HL 和 BC 减 1
ED B9	‘CPDR’-HL 和 BC 减 1，重复直到 BC = 0 或找到需要的字符

HL　指示其内容应与累加器内容相比较的存储单元地址
BC　字节计数器

交换指令组和数据块传送和查找指令组

助记符	用符号表示的操作	标志 S	Z		H		P/V	N	C	操作码 76 543 210	Hex	字节数	M周期数	T状态数	说明
EXDE, HL	DE↔HL	•	•	X	•	X	•	•	•	11 101 011	E B	1	1	4	
EXAF, AF′	AF↔AF′	•	•	X	•	X	•	•	•	11 011 000	D8	1	1	4	
EXX	(BC↔BC′ DE↔DE′ HL↔HL′)	•	•	X	•	X	•	•	•	11 011 001	D9	1	1	4	寄存器组和辅助寄存器组交换
EX(SP),HL	H↔(SP+1) L↔(SP)	•	•	X	•	X	•	•	•	11 100 011	E3	1	5	19	
EX(SP), IX	IX_H↔(SP+1) IX_L↔(SP)	•	•	X	•	X	•	•	•	11 011 101 11 100 011	D D E3	2	6	23	
EX(SP), IY	IY_H↔(SP+1) IY_L↔(SP)	•	•	X	•	X	•	•	•	11 111 101 11 100 011	F D E3	2	6	23	
LDI	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC−1	•	•	X	0	X	① ↕	•	•	11 101 101 10 100 000	E D A0	2	4	16	(HL)送到(DE)，指针增1，字节计数器(BC)减1
LDIR	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC−1 重复直到BC=0	•	•	X	0	X	0	0	•	11 101 101 10 110 000	E D B D	2 2	5 4	21 16	如 BC≠0 如 BC=0
LDD	(DE)←(HL)	•	•	X	0	X	① ↕	0	•	11 101 101 10 101 000	E D A8	2	4	16	
LDDR	(DE)←(HL) DE←DE−1 HL←HL−1 BC←BC−1 重复直到BC=0	•	•	X	0	X	0	0	•	11 101 101 10 111 000	E D B8	2 2	5 4	21 16	如 BC≠0 如 BC=0
			②												

续表

助记符	用符号表示的操作	标志								操作码		字节数	M周期数	T状态数	说明
		S	Z		H		P/V	N	C	76 543 210	Hex				
CPI	A－(HL) HL←HL＋1 BC←BC－1	↕	↕	X	↕	X	↕	1	•	11 101 101 10 100 001	E D A 1	2	4	16	
CPIR	A－(HL) HL←HL＋1 BC←BC－1 重复直到 A＝(HL)或 BC＝0	↕	②↕	X	↕	X	①↕	1	•	11 101 101 10 110 001	E D B 1	2 2	5 4	21 16	如 BC≠0 和 A≠(HL) 如 BC＝0 或 A＝(HL)
CPD	A－(HL) HL←HL－1 BC←BC－1	↕	②↕	X	↕	X	①↕	1	•	11 101 101 10 101 001	E D A 9	2	4	16	
CPDR	A－(HL) HL←HL－1 BC→BC－1 重复直到 A＝(HL)或BC＝0	↕	②↕	X	↕	X	①↕	1	•	11 101 101 10 111 001	ED B9	2 2	5 4	21 16	如 BC≠0 和 A≠(HL) 如 BC＝0 或 A＝(HL)

注：①如BC－1的结果为0；P/V 标志为0，否则 P/V＝1，②如A＝(HL)，Z标志为1，否则Z＝0

标志记号同前

8位算术、逻辑指令组

源

	寄存器寻址							寄存器间接	变址		立即
	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	n
加 (ADD)	87	80	81	82	83	84	85	86	DD 86 d	FD 86 d	C6 n
带进位加 (ADC)	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n
减 (SUB)	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n
带进位减 (SBC)	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n
'与' (AND)	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n
'异' (XOR)	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n
'或' (OR)	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n
比较 (CP)	BF	B8	B9	BA	BB	BE	BD	BE	DD BE d	FD BE d	FE n
增量 (INC)	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d	
减量 (DEC)	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d	

8位算术、逻辑指令组

助记符	用符号表示的操作	S	Z		H		P/V	N	C	76	543	210	Hex	字节数	M周期数	T状态数	说明	
ADDA.r	A←A+r	↕	↕	X	↕	X	V	0	↕	10	[000]	r		1	1	4	r	寄存器
ADDA.n	A←A+n	↕	↕	X	↕	X	V	0	↕	11	[000]	110		2	2	7	000	B
										←	n	→					001	C
																	010	D
ADDA.(HL)	A←A+(HL)	↕	↕	X	↕	X	V	0	↕	10	[000]	110		1	2	7	011	E
ADDA.(IX+d)	A←A+(IX+d)	↕	↕	X	↕	X	V	0	↕	11	011	101	DD	3	5	19	100	H
										10	[000]	110					101	L
										←	d	→					111	A
ADDA.(IY+d)	A←A+(IY+d)	↕	↕	X	↕	X	V	0	↕	11	111	101	FD	3	5	19		
										10	[000]	110						
										←	d	→						
ADCA.s	A←A+s+CY	↕	↕	X	↕	X	V	0	↕		[001]						同ADD指令那样，S为r、n (HL)、(IX+d)、(IY+d)中的任意一个，用方框所标出的位代替ADD指令系列中的000	
AUB.S	A←A−S	↕	↕	X		X	V	1	↕		[010]							
SBCA.S	A←A−S−CY	↕	↕	X		X	V	1	↕		[011]							
ANDS	A←A∧S	↕	↕	X	1	X	P	0	0		[100]							
ORS	A←A∨S	↕	↕	X	0	X	P	0	0		[110]							
XORS	A←A⊕s	↕	↕	X	0	X	P	0	0		[101]							
CPS	A−S	↕	↕	X	↕	X	V	1	↕		[111]							
INCr	r←r+1	↕	↕	X	↕	X	V	0	·	00	r	[100]		1	1	4		
INC(HL)	(HL)←(HL)+1	↕	↕	X	↕	X	V	0	·	00	110	[100]		1	3	11		
INC(IX+d)	(IX+d)←(IX+d)+1	↕	↕	X	↕	X	V	0	·	11	011	101	DD	3	6	23		
										00	110	[100]						
										←	d	→						
INC(IY+d)	(IY+d)←(IY+d)+1	↕	↕	X	↕	X	V	0	·	11	111	101	FD	3	6	23		
										00	110	[100]						
										←	d	→						
DF.CS	S←S−1	↕	↕	X	↕	X	V	1	·			[101]					象INC指令那样，S是r (HL) (IX+d)、(IY+d)中的任意一个，DEC指令的格式和状态同INC，在操作码中用101代替100	

注：P/V标志一列中的V符号指明，P/V标志包含着运算结果的溢出情况，类似地，P符号指明奇偶性，V=1表示溢出，V=0表示不溢出，P=1表示结果为偶数，P=0表示结果为奇。数标志记号同前。

通 用 运 算 指 令

十进制调整累加器(DAA)	27
累加器变反(CPL)	2F
累加器变补(NEG)(2的补码)	ED 44
进位标志变反(CCF)	3F
置位进位标志(SCF)	37

其它 CPU 控制指令

不操作(NOP)	00	
暂停(HALT)	76	
禁止中断(DI)	F3	
允许中断(EI)	FB	
置中断方式 0(IM0)	ED 46	8080A 方式
置中断方式 1(IM1)	ED 56	转向 0038H 单元重新启动
置中断方式 2(IM2)	ED 5E	利用寄存器 I 的内容和请求中断的设备提供的 8 位作指针的间接调用

通用算术　CPU 控制指令组

助记符	用符号表示的操作	标志 S	Z		H		P/V	N	C	操作码 76 543 210	Hex	字节数	M周期数	T状态数	说明
DAA	此指令随着BCD操作数的加法或减法指令后面，转换累加器内容成BCD形式	↕	↕	X	↕	X	P	•	↕	00 100 111	27	1	1	4	十进制调整累加器
CPL	A←$\overline{A}$	•	•	X	1	X	•	1	•	00 101 111	2F	1	1	4	累加器变反（求1的补码）
NEG	A←$\overline{A}$+1	↕	↕	X	↕	X	V	1	↕	11 101 101	E D	2	2	8	累加器变补（求2的补码）
										01 000 100	44				
CCF	CY←$\overline{CY}$	•	•	X	X	X	•	0	↕	00 111 111	3F	1	1	4	进位标志变反
SCF	CY←1	•	•	X	0	X	•	0	1	00 110 111	37	1	1	4	进位标志置位
NOP	不操作	•	•	X	•	X	•	•	•	00 000 000	00	1	1	4	
HALT	CPU 暂停	•	•	X	•	X	•	•	•	01 110 110	76	1	1	4	
DI*	IFF←0	•	•	X	•	X	•	•	•	11 110 011	F3	1	1	4	
EI*	IFF→0	•	•	X	•	X	•	•	•	11 111 011	FB	1	1	4	
IMO	置中断方式0	•	•	X	•	X	•	•	•	11 101 101	E D	2	2	8	
										01 000 110	46				
IM1	置中断方式1	•	•	X	•	X	•	•	•	11 101 101	E D	2	2	8	
										01 010 110	56				
IM2	置中断方式2	•	•	X	•	X	•	•	•	11 101 101	E D	2	2	8	
										01 011 110	5E				

注：IFF 表示中断允许触发器

CY 表示进位位触发器

标志记号同前，*=EI 和 DI 指令结束时，并取样中断信号

16位 算 术 指 令

目的地			源 BC	ED	HL	XP	IX	IY
	加(ADD)	HL	09	19	29	39		
	加(ADD)	IX	DD 09	DD 19		DD 39	DD 29	
	加(ADD)	IY	FD 09	FD 19		FD 39		FD 29
	带进位加和置位标志(ADC)	HL	ED 4A	ED 5A	ED 6A	ED 7A		
	带进位减和置位标志(SBC)	HL	ED 42	ED 52	ED 62	ED 72		
	增量(INC)		03	13	23	33	DD 23	FD 23
	减量(DEC)		0B	1B	2B	3B	DD 2B	FD 2B

16位算术指令组

助记符	用符号表示的操作	标					志			操作码		字节数	M周期数	T状态数	说明
		S	Z		H		P/V	N	C	76 543 210	Hex				
ADD HL,ss	HL←HL+ss	•	•	X	X	X	•	0	↕	00 ss1 001		1	3	11	rr 寄存器
															00 BC
ADC HK,ss	HL→ HL+ss+CY	↕	↕	X	X	X	V	0	↕	11 101 101	E D	2	4	15	01 DE
										01 ss1 010					10 HL
															11 SP
SBC HL,ss	HL← HL−ss−CY	↕	↕	X	X	X	V	1	↕	11 101 101	E D	2	4	15	
										01 ss0 010					
ADD IX,pp	IX←IX+pp	•	•	X	X	X	•	0	↕	11 011 101	D D	2	4	15	qq 寄存器
										00 pp1 001					00 BC
															01 DE
															10 IX
															11 SP
ADD IY,rr	IY←IY+rr	•	•	X	X	X	•	0	↕	11 111 101	F D	2	4	15	rr 寄存器
										00 rr1 001					00 BC
															01 DE
															10 IY
															11 Sp
INCss	ss←ss+1	•	•	X	•	X	•	•	•	00 ss0 011		1	1	6	
INC IX	IX←IX+1	•	•	X	•	X	•	•	•	11 011 101	D D	2	2	10	
										00 100 011	23				
INC IY	IY←IY+1	•	•	X	•	X	•	•	•	11 111 101	F D	2	2	10	
										00 100 011	23				
DECss	ss←ss−1	•	•	X	•	X	•	•	•	00 ss1 011		1	1	6	
DEC IX	IX←IX−1	•	•	X	•	X	•	•	•	11 011 101	D D	2	2	10	
										00 101 011	2B				
DEC IY	IY←IY−1	•	•	X	•	X	•	•	•	11 111 101	F D	2	2	10	
										00 101 101	2B				

注：ss是寄存器对BC、DE、HL、SP中的任意一个

pp是寄存器对BC、DE、HL、SP中的任意一个

rr存器对BC、DE、HL、SP是寄中的任意一个

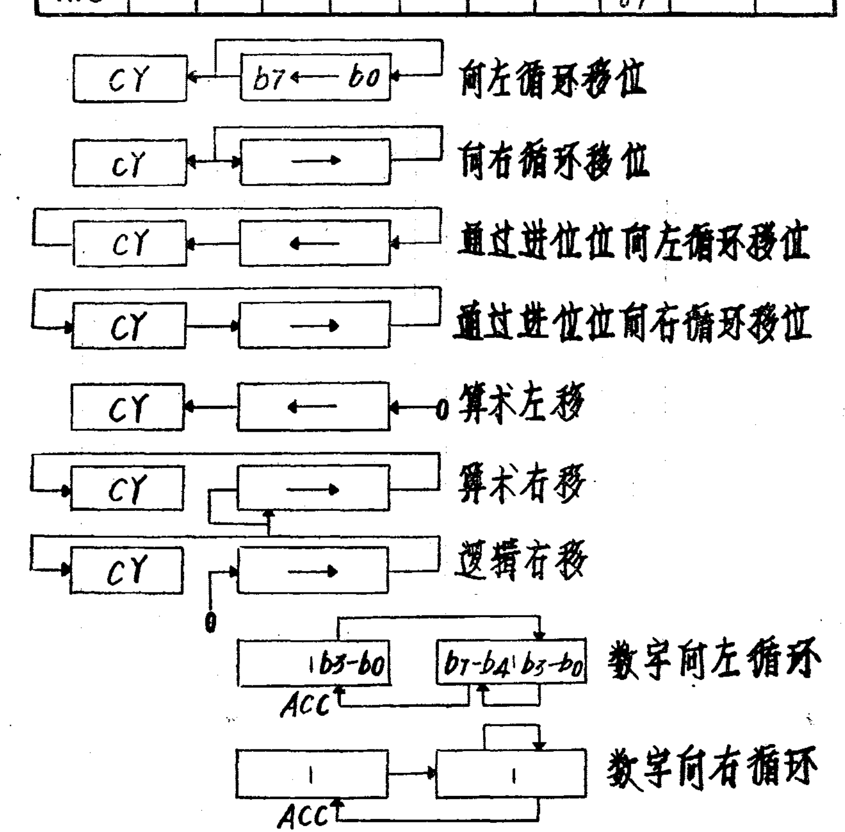
标志记号同前

循 环 和 移 位 指 令

源和目的地

循环或移位的类型	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)
'RLC'	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06	DD CB d 06	FD CB d 06
'RRC'	CB 0F	CB 08	CB 09	CB 0A	CB 0B	CB 0C	CB 0D	CB 0E	DD CB d 0E	FD CB d 0E
'RL'	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16
'RR'	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E
'SLA'	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26
'SRA'	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E
'SRL'	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB d 3E	FD CB d 3E
'RLD'								ED 6F		
'RRD'								ED 67		

	A
'RLCA'	07
'RRCA'	0F
'RLA'	17
'RRA'	1F



循环和移位

助记符	用符号表示的操作	S	Z		H		P/V	N	C	75 543 210	Hex	字节数	M周期数	T状态数	说明
RLCA	CY ← [7←0] ← (循环) A	·	·	X	0	X	·	0	↕	00 000 111	07	1	1	4	累加器向左循环移位
RLA	[CY] ← [7←0] ← (循环) A	·	·	C	0	X	·	0	↕	00 010 111	17	1	1	4	累加器向左循环移位通过进位位
RRCA	[7→0] → CY (循环) A	·	·	X	0	X	·	0	↕	00 001 111	0F	1	1	4	累加器向右循环移位
RRA	[7→0] → [CY] (循环) A	·	·	X	0	X	·	0	↕	00 011 111	1F	1	1	4	累加器向右循环移位通过进位位
RLCr	CY ← [7←0] (循环) r.(HL).(IX+d).(IY+d)	↕	↕	X	0	X	P	0	↕	11 011 011 00 [000] r	CB	2	2	8	寄存器r向左循环移位
RLC (HL)		↕	↕	X	0	X	P	0	↕	11 001 011 00 [000] 110	CB	2	4	15	r 寄存器 000 B 001 C
RLC (IX+d)		↕	↕	X	0	X	P	0	↕	11 011 101 11 001 011 ← d → 00 [000] 110	DD CB	4	6	23	010 D 011 E 100 H 101 L 111 A
RLC (IY+d)		↕	↕	X	0	X	P	0	↕	11 111 101 11 001 011 ← d → 00 [000] 110	FD CB	4	6	23	
RLs	[CY] ← [7←0] ← (循环) s≡r.(HL).(IX+d).(IY+d)	↕	↕	X	0	X	P	0	↕	[010]					指令格式和状态周期与RLC s相同形成新的操作码，用新操作码代替RLC中的[000]
RRCs	[7→0] → CY (循环) s≡r.(HL).(IX+d).(IY+d)	↕	↕	X	0	X	P	0	↕	[001]					
RRs	[7→0] → [CY] (循环) s≡r.(HL).(IX+d).(IY+d)	↕	↕	X	0	X	P	0	↕	[011]					
SLAs	CY ← [7←0] ← 0 s≡r.(HL).(IX+d).(IY+d)	↕	↕	X	0	X	P	0	↕	[100]					
SRAs	[7→0] → CY (7位保持) s≡r.(HL).(IX+d).(IY+d)	↕	↕	X	0	X	P	0	↕	[101]					
SRLs	0 → [7→0] → CY s≡r.(HL).(IX+d).(IY+d)	↕	↕	X	0	X	P	0	↕	[111]					
RLD	A [7-4 | 3-0] [7-4 | 3-0](HL)	↕	↕	X	0	X	P	0	·	11 101 101 01 101 101	ED 6F	2	5	18	BCD数字在累加器和存储单元(HL)间向左和向右循环移位
RRD	A [7-4 | 3-0] [7-4 | 3-0](HL)	↕	↕	X	0	X	P	0	·	11 101 101 01 100 111	ED 67	2	5	18	累加器高四位内容不受影响.

标志记号同前

位 操 作 指 令 组

	位	寄存器寻址							寄存器接间	变址	
		A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)
位测试(BIT)	0	CB 47	CB 40	CB 41	CB 42	CB 43	CB 44	CB 45	CB 46	DD CB d 46	FD CB d 46
	1	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E	DD CB d 4E	FD CB D 4E
	2	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56	DD CB d 56	FD CB d 56
	3	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E	DD CB d 5E	FD CB d 5E
	4	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66	DD CB d 66	FD CB d 66
	5	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	CB 6E	DD CB d 6E	FD CB d 6E
	6	CB 77	CB 70	CB 71	CB 72	CB 73	CB 74	CB 75	CB 76	DD CB d 76	FD CB d 76
	7	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	CB 7E	DD CB d 7E	FD CB d 7E
位复位(RES)	0	CB 87	CB 80	CB 81	CB 82	CB 83	CB 84	CB 85	CB 86	DD CB d 86	FD CB d 86
	1	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	CB 8E	DD CB d 8E	FD CB d 8E
	2	CB 97	CB 90	CB 91	CB 92	CB 93	CB 94	CB 95	CB 96	DD CB d 96	FD CB d 96
	3	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	CB 9E	DD CB d 9E	FD CB d 9E

位		寄存器寻址							寄存器接间	变址	
		A	B	C	D	E	H	L	(HL)	(IX + d)	(IY+d)
位复位(RES)	4	CB A7	CB A0	CB A1	CB A2	CB A3	CB A4	CB A5	CB A6	DD CB d A6	FD CB d A6
	5	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	CB AE	DD CB d AE	FD CB d AE
	6	CB B7	CB B0	CB B1	CB B2	CB B3	CB B4	CB B5	CB B6	DD CB d B6	FD CB d B6
	7	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	CB BE	DD CB d BE	FD CB d BE
位置位(SET)	0	CB C7	CB C0	CB C1	CB C1	CB C3	CB C4	CB C5	CB C6	DD CB d C6	FD CB d C6
	1	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	CB CE	DD CB d CE	FD CB d CE
	2	CB D7	CB D0	CB D1	CB D2	CB D3	CB D4	CB D5	CB D6	DD CB d D6	DF CB d D6
	3	CB DF	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	CB DE	DD CB d DE	FD CB d DE
	4	CB E7	CB E0	CB E1	CB E2	CB E3	CB E4	CB E5	CB E6	DD CB d E6	FD CB d E6
	5	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	CB EE	DD CB d EE	FD CB d EE
	6	CB F7	CB F0	CB F1	CB F2	CB F3	CB F4	CB F5	CB F6	DD CB d F6	FD CB d F6
	7	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	CB FE	DD CB d FE	FD CB d FE

位操作(置位、复位和测试)指令组

助记符	用符号表示的操作	标志 S	Z		H		P/V	N	C	操作码 76 543 210	Hex	字节数	M周期数	T状态数
BIT b，r	$Z \leftarrow \overline{r_b}$	X	↕	X	1	X	X	0	•	11 001 011	C B	2	2	8
										01 b r				
BIT b，(HL)	$Z \leftarrow \overline{(HL)_b}$	X	↕	X	1	X	X	0	•	11 001 011	C B	2	3	12
										01 b 110				
BIT b，$(IX+d)_b$	$Z \leftarrow \overline{(IX+d)_b}$	X	↕	X	1	X	X	0	•	11 011 101	D D	4	5	20
										11 001 011	C B			
										← d →				
										01 b 110				
BIT b，$(IY+d)_b$	$Z \leftarrow \overline{(IY+d)_b}$	X	↕	X	1	X	X	0	•	11 111 101	F D	4	5	20
										11 001 011	C B			
										← d →				
										01 b 110				
SET b，r	$r_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 001 011	C B	2	2	8
										[11] b r				
SET b，(HL)	$(HL)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 001 011	C B	2	4	15

说明：

r	寄存器
000	B
001	C
010	D
011	E
100	H
101	L
111	A

b	被测位
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

助记符	用符号表示的操作	S	Z		H		P/V	N	C	操作码 76 543 210	Hex	字节数	M周期数	T状态数	说明
SETb, (IX+d)	$(IX+d)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 011 101 11 001 011 ← d → [11] b 110	DD CB	4	6	23	
SETb, (IY+d)	$(IY+d)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 111 101 11 001 011 ← d → [11] b 110	FD CB	4	6	23	
RESb, s	$S_b \leftarrow 0$ S=r, (HL), (IX+d), (IY+d),	•	•	X	•	X	•	•	•	[10]					为形成新的操作码，用[10]代替置位指令SET_b，S中的[11]标志和时间状态同SET指令

注：记号S_b指明存储单元S的第b位(b=0～7)
标志记号同前

转 移 指 令 组

条　　件

			无条件	进位	无进位	零	非零	校验偶	校验奇	符号为负	符号为正	寄存器 B≠0
转移(JP)	立即扩展	n n	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
转移(JP)	相　　对	PC+e	18 e-2	38 e-2	30 e-2	28 e-2	20 e-2					
转移(JP)	寄 存 器	(HL)	E9									
转移(JP)	间　　接	(IX)	DD E9									
转移(JP)		(IY)	FD E9									
B减1,非零转 (DJNZ)	相　　对	PC+e										10 e-2

转移指令组

助记符	用符号表示的操作	S	Z		H		P/V	N	C	操作码 76 543 210	Hex	字节数	M周期数	T状态数	说明
JPnn	PC←nn	•	•	X	•	X	•	•	•	11 000 011	C3	3	3	10	
										← n →					
										← n →					
JPcc,nn	若条件cc是真，PC←nn，否则继续	•	•	X	•	X	•	•	•	11 cc 010		3	3	10	cc 条件
										← n →					000 NZ非零
										← n →					001 Z 零
															010 NC无进位
															011 C 进位
															100 PO奇偶奇
															101 PE奇偶偶
															110 P 正号
JRe	PC←PC+e	•	•	X	•	X	•	•	•	00 011 000	18	2	3	12	111 M 负号
										← e−2 →					
JRC,e	若C=0，则继续	•	•	X	•	X	•	•	•	00 111 000	38	2	2	7	如果条件不满足
	若C=1,PC←PC+e									← e−2 →					
												2	3	12	如果条件满足
JRNC,e	若C=1	•	•	X	•	X	•	•	•	00 110 000	30	2	2	7	如果条件不满足
	则继续									← e−2 →					
	若C=0,PC←PC+e											2	3	12	如果条件满足

续表

助别符	用符号表示的操作	标志 S	Z		H		P/V	N	C	操作码 76 543 210	Hex	字节数	M周期数	T状态数	说明
JRZ,e	若Z=0	•	•	X	•	X	•	•	•	00 101 000	28	2	2	7	如果条件不满足
	则继续	•	•	X	•	X	•	•	•	← e−2 →					
	若Z=1,PC←PC+e											2	3	12	如电条件满足
JRNZ, e	若Z=1,	•	•	X	•	X	•	•	•	00 100 000	20	2	2	7	如果条件不满足
	则继续									← e −2 →					
	若Z=0,PC←PC+e											2	3	12	如果条件满足
JP(HL)	PC←HL	•	•	X	•	X	•	•	•	11 101 001	E9	1	1	4	
JP(IX)	PC←IX	•	•	X	•	X	•	•	•	11 011 101	DD	2	2	8	
										11 101 001	E9				
JP(IY)	PC←IY	•	•	X	•	X	•	•	•	11 111 101	FD	2	2	8	
										11 101 001	E9				
DJNZe	B←B−1	•	•	X	•	X	•	•	•	00 010 000	10	2	2	8	若B=0
	若B=0，则继续									← e −2 →					
	若B≠0,											2	3	13	若B≠0
	PC←PC+e														

注：e表示相对寻址方式中地址的偏移量。e为带正负号的2的补码表示的数，其范围为(−126, 129)。操作码中的e − 2提供有效性地址PC+e，因为在加e之前PC已经加2

标志同前

调用和返回指令组

		条件	无条件	进位	无进位	零	非零	校验偶	校验奇	符号为负	符号为正	寄存器 B≠0
调用 (CALL)	立即扩展	nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
返回 (RET)	寄存器间接	(SP) (SP+1)	C9	D8	D0	C8	C0	E8	E0	F8	F0	—
从中断返回 (RETI)	寄存器间接	(SP) (SP+1)	ED 4D									
从不可屏蔽中断返回(RETN)	寄存器间接	(SP) (SP+1)	ED 45									

注：某些标志具有多种用途。详见 Z80-CPU 技术手册

重新启动指令组

		操作码	
调来地址	0000H	C7	‘RST0’
	0008H	CF	‘RST8’
	0010H	D7	‘DST16’
	0018H	DF	‘RST24’
	0020H	E7	‘RST32’
	0028H	EF	‘RST40’
	0030H	F7	‘RST48’
	0038H	FF	‘RST56’

调 用 和 返 回 指 令 组

助记符	用符号表示的操作	标志 S	Z		H		P/V	N	C	操作码 76 543 210	Hex	字节数	M周期数	T状态数	说明
CALLnn	(SP−1)←PC_H	•	•	X	•	X	•	•	•	11 001 101	CD	3	5	17	
	(SP−2)←PC_L									← n →					
	PC←nn									← n →					
CALLcc,nn	若条件 cc 是假，则继	•	•	X	•	X	•	•	•	11 cc 100		3	3	10	若 cc 是假
	续；否则，同									← n →		3	5	17	若 cc 是真
	CALLnn														
RET	PC_L←(SP)	•	•	X	•	X	•	•	•	11 001 001	C9	1	3	10	
	PC_H←(SP+1)														
RETcc	若条件 cc 是假则继续	•	•	X	•	X	•	•	•	11 cc 000		1	1	5	若 cc 是假
	否则，同 RET											1	3	11	若 cc 是真
RETI	从中断返回	•	•	X	•	X	•	•	•	11 101 101	ED	2	4	14	
										01 001 101	4D				
RETN	从中不可屏蔽	•	•	X	•	X	•	•	•	11 101 101	ED	2	4	14	
	中断返回									01 000 101	45				

cc	条件
000	NZ 非零
001	Z 零
010	NC无进位
011	C 进位
100	PO奇偶奇
101	PE奇偶偶
110	P 正号
111	M 负号

续表

助记符	用符号表示的操作	标志								操作码				字节数	M周期数	T状态数	说明
		S	Z		H		P/V	N	C	76	543	210	Hex				
RSP_P	$(SP-1) \leftarrow PC_L$ $(SP-2) \leftarrow PC_L$ $PC_H \leftarrow 0$ $PC_L \leftarrow P$	•	•	X	•	X	•	•	•	11	t	111		1	3	11	

t	p
000	00_H
001	08_H
010	10_H
011	18_H
100	20_H
101	28_H
110	30_H
111	38_H

[1]RETN 指令将 IFF_2 传送到 $IFF_1(IFF_2 \rightarrow IFF_1)$

标志记号同前

输 入 指 令 组

输入目的地

			口地址：立即 n	口地址：存存器间接 (c)
输入(IN)	寄存器寻址	A	DB n	ED 78
		B		ED 40
		C		ED 48
		D		ED 50
		E		ED 58
		H		ED 60
		L		ED 68
输入，HL增1，B减1(INI)	寄存器间接	(HL)		ED A2
输入,HL增1,B减1如B≑0则重复(INIR)				ED B2
输入，HL减1，B减1(IND)				ED AA
输入，HL减1,B减1,如B≑0则重复(INDR)				ED BA

（INI、INIR、IND、INDR：数据块输入指令）

输 出 指 令 组

源

	口目的地址		存存器 A	B	C	D	E	H	L	存存器间接 (HL)
输出(OUT)	立即	n	D3 n							
	寄存器间接	(c)	E D 79	E D 41	E D 49	E D 51	E D 59	E D 61	E D 69	
输出，HL增1，B减1(OUTI)	寄存器间接	(c)								E D A3
输出，HL增1，B减1，如B≑0则重复(OTIR)	寄存器间接	(c)								E D B3
输出，HL增1，B减1，(OUTD)	寄存器间接	(c)								E D A B
输出,HL减1,B减1，如B≑0则重复(DTDR)	寄存器间接	(c)								E D B B

（OUTI、OTIR、OUTD、DTDR：数据块输出指令）

输 入 和 输 出

助记符	用符号表示的操作	标志 S	Z		H		P/V	N	C	操作码 76 543 210	Hex	字节数	M周期数	T状态数	说明
INA, (n)	A→(n)	•	•	X	•	X	•	•	•	11 011 011 ← n →	DB	2	3	11	n至 $A_0 \sim A_7$ 累加器至 $A_8 \sim A_{15}$
INr, (c)	r→(c) 若r = 110, 仅标志受影响	↕	↕	X	↕	X	P	0	•	11 101 101 01 r 000	ED	2	3	12	C至 $A_0 \sim A_7$ B至 $A_8 \sim A_{15}$
INI	(HL)←C B←B−1 HL←HL+1	X	↕	① X	X	X	X	1	•	11 101 101 10 100 101	ED A2	2	4	16	C至 $A_0 \sim A_7$ B至 $A_8 \sim A_{15}$
INIR	(HL)←(c) B←B−1 HL←HL+1 一直重复到B = 0	X	1	X	X	X	X	1	•	11 101 101 10 110 010	ED B2	2 2	5 (若B≠0) 4 (若B = 0)	21 16	C至 $A_0 \sim A_7$ B至 $A_8 \sim A_{15}$
IND	(HL)←(C) B←B−1 HL←HL−1	X	① ↕	X	X	X	X	1	•	11 101 101 10 111 010	ED AA	2	4	16	C至 $A_0 \sim A_7$ B至 $A_8 \sim A_{15}$
INDR	(HL)←C B←B−1 HL←HL−1 一直重复到B = 0	X	1	X	X	X	X	1	•	11 101 101 10 111 010	ED	2 2	5 (若B≠0) 4 (若B = 0)	21 16	C至 $A_0 \sim A_7$ B至 $A_8 \sim A_{15}$
OUT(n), A	(n)←A	•	•	X	•	X	•	•	•	11 010 011 ← n →	D3	2	3	11	n至 $A_0 \sim A_7$ 累加器至 $A_8 \sim A_{15}$
OUT(c), r	(C)←r	•	• ①	X	•	X	•	•	•	11 101 101 01 r 011	ED	2	3	12	C至 $A_0 \sim A_7$ B至 $A_8 \sim A_{15}$

续表

助记符	用符号表示的操作	标志 S	Z		H		P/V	N	C	操作码 76 543 210	Hex	字节数	M周期数	T状态数	说明
OUTI	(C)←(HL) B←B－1 HL←HL＋1	X	↕	X	X	X	X	1	•	11 101 101 10 100 011	ED A3	2	4	16	C至 A_0～A_7 B至 A_8～A_{15}
OTIR	(C)←(HL) B←B－1 HL←HL＋1 一直重复到B＝0	X	1	X	X	X	X	1	•	11 101 101 10 110 011	ED B3	2 2	5 (若B≠0) 4 (若B＝0)	21 16	C至 A_0～A_7 B至 A_8～A_{15}
OUTD	(C)←(HL) B←B－1 HL←HL－1	X	① ↕	X	X	X	X	1	•	11 101 101 10 101 011	ED AB	2	4	16	C至 A_0～A_7 B至 A_8～A_{15}
OTDR	(C)←(HL) B←B－1 HL←HL－1 一直重复到B＝0	X	1	X	X	X	X	1	•	11 101 101 10 111 011	ED BB	2 2	(若B≠0) 4 (若B＝0)	21 16	C至 A_0～A_7 B至 A_8～A_{15}

注：① 若B－1的结果为0，P/V标志置1，否则置0
标志记号同前

Z-80CPU 中断结构

可屏蔽中断($\overline{INT}$)

方式 0

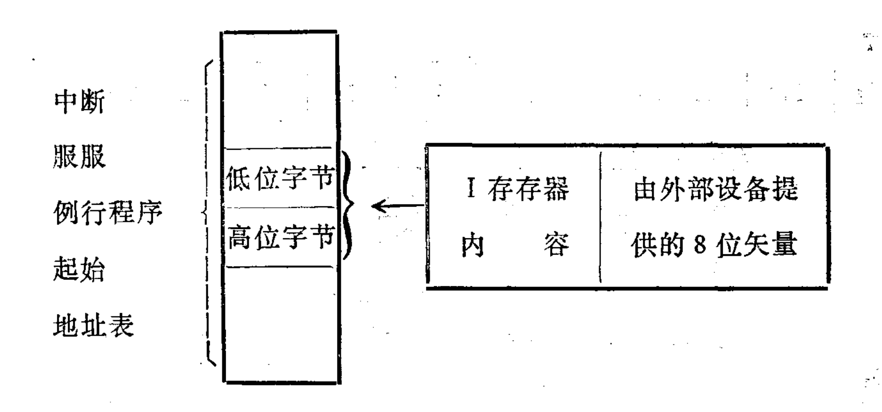
如同 8080A 的中断响应方式，在$\overline{INTA}=\overline{MI}\cdot\overline{IORQ}$期间将指令放到数据总线上。

方式 1

重新启动转到 38_H 或 56_{10}(RST_{56})单元。

方式 2

由 Z80 外部设备提供中断矢量。



不可屏蔽中断($\overline{NMI}$)

重新启动转到 66_H 或 102_{10} 单元

中断允许/禁止触发器

动作	IFF_1	IFF_2	
CPU 复位	0	0	
DI	0	0	
EI	1	1	
LDA.I	•	•	IFF_1→奇偶标志
LDA.R	•	•	IFF_2→奇偶标志
接受$\overline{NMI}$	0	•	
不可屏蔽中断返回	IFF_2	•	IFF_2→IFF_1
接受$\overline{INT}$	0	0	
中断返回	•	•	

“•”表示无变化

PIO 编 程 摘 要

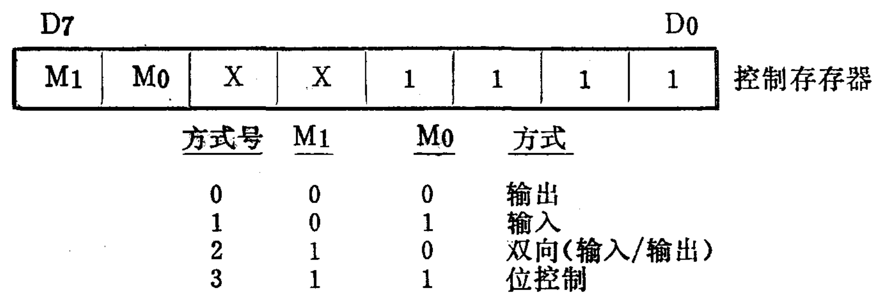
存存器选择

选择线 C/D	选择线 B/A	所选中存存器
0	0	A 口数据字
0	1	B 口数据字
1	0	A 口控制字
1	1	B 口控制字

装入中断矢量

D7							D0	
V7	V6	V5	V4	V3	V2	V1	V0	控制存存器

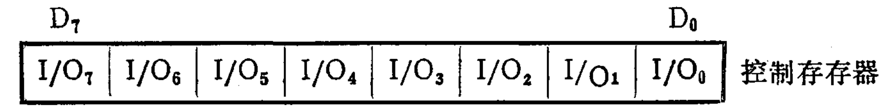
置工作方式



D7							D0	
M1	M0	X	X	1	1	1	1	控制存存器

方式号	M1	M0	方式
0	0	0	输出
1	0	1	输入
2	1	0	双向(输入/输出)
3	1	1	位控制

如果选中方式 3，送向 PIO 的下一个控制字是

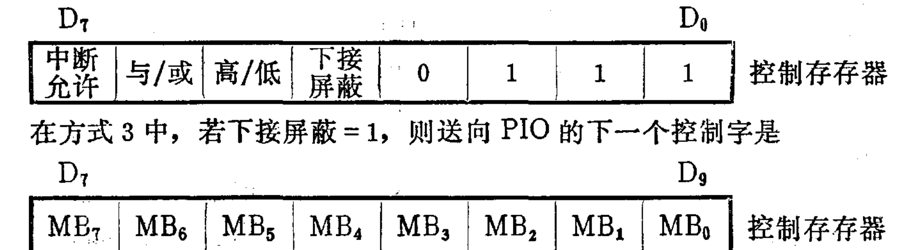


D_7							D_0	
I/O_7	I/O_6	I/O_5	I/O_4	I/O_3	I/O_2	I/O_1	I/O_0	控制存存器

I/O 置“1”位用于输入

I/O 置“0”位用于输出

置中断控制



D_7							D_0	
中断允许	与/或	高/低	下接屏蔽	0	1	1	1	控制存存器

在方式 3 中，若下接屏蔽 = 1，则送向 PIO 的下一个控制字是

D_7							D_9	
MB_7	MB_6	MB_5	MB_4	MB_3	MB_2	MB_1	MB_0	控制存存器

MB = 0 监控此位 MB = 1，屏蔽此位

允许/禁止中断

D_0							D_0	
中断允许	X	X	X	0	0	1	1	控制寄存器

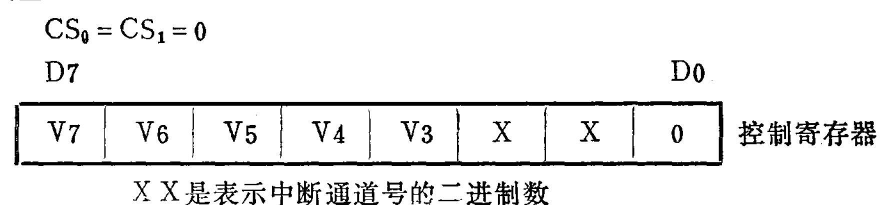
CTC编 程 摘 要

寄存器选择

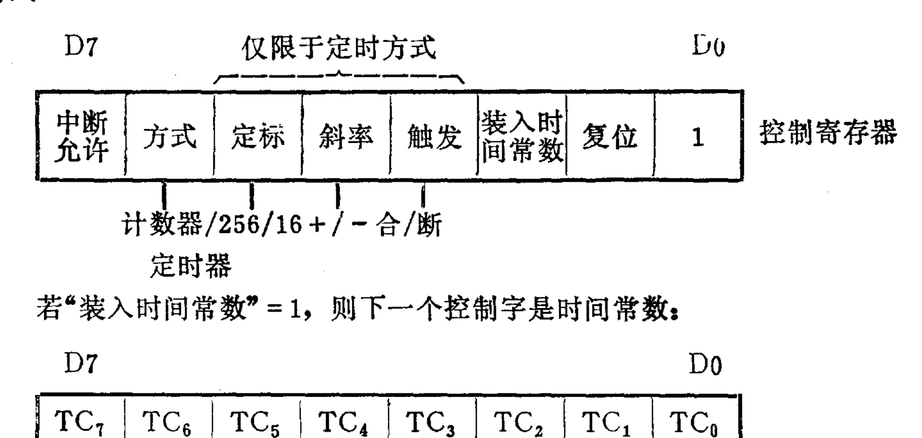
选择线 CS_1	选择线 CS_0	通道选择	优先级
0	0	0	最高级
0	1	1	
1	0	2	
1	1	3	最低级

“读” = 减 1 计数器　　“写” = 控制寄存器

装入中断矢量



置工作方式



若“装入时间常数” = 1，则下一个控制字是时间常数：

D7　　D0

TC_7	TC_6	TC_5	TC_4	TC_3	TC_2	TC_1	TC_0

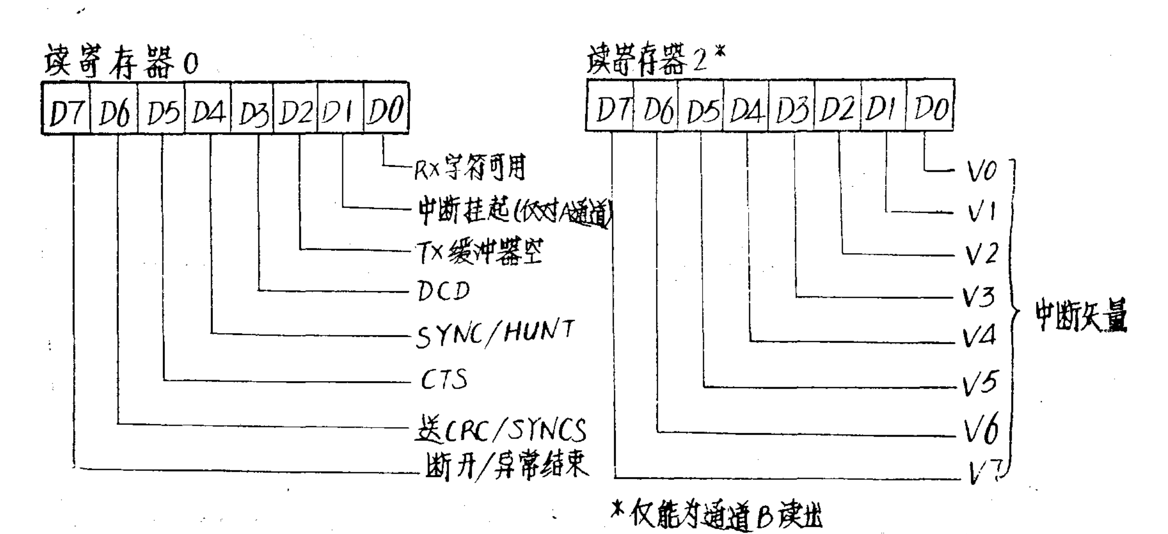
当 01_H 减至 00_H 时，CTC 通道中断

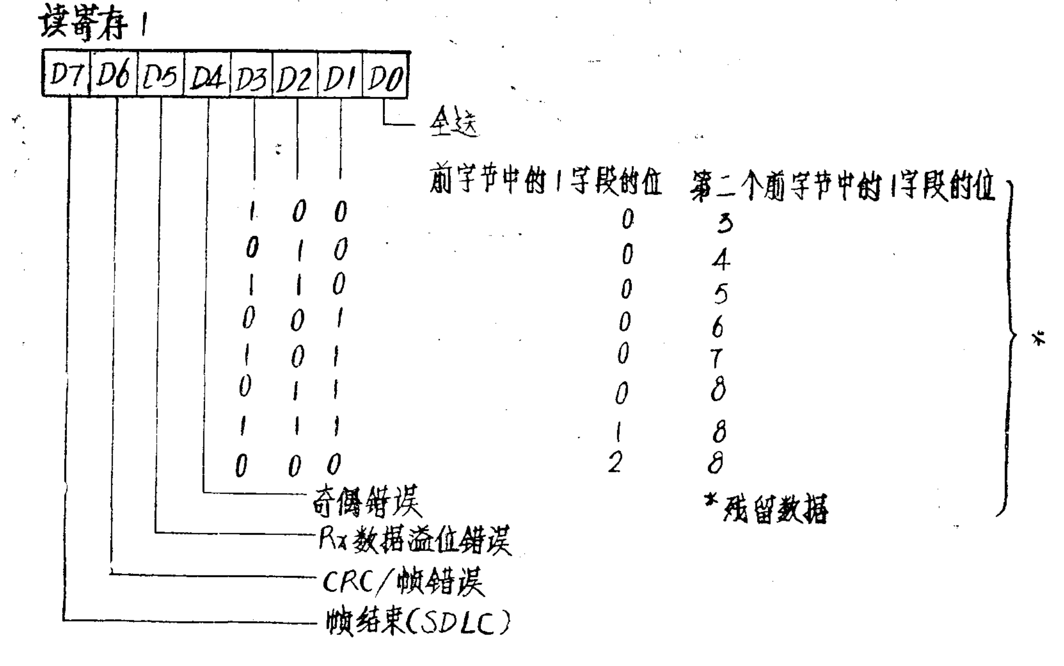
时间量	到中断的十进制计数
01_H	1
•	•
•	•
FF_H	525
00_H	256

SIO 编 程 摘 要

通 道 选 择

C/D	B/A	功　　　　能
0	0	通道A数据
0	1	通道B数据
1	0	通道A命令/状态
1	1	通道B命令/状态





SIO 编 程 摘 要(续)

写 寄 存 器

写寄存器 0

D7	D6	D5	D4	D3	D2	D1	D0

D2	D1	D0	
0	0	0	寄存器0
0	0	1	寄存器1
0	1	0	寄存器2
0	1	1	寄存器3
1	0	0	寄存器4
1	0	1	寄存器5
1	1	0	寄存器6
1	1	1	寄存器7

D5	D4	D3	
0	0	0	无效代码
0	0	1	送异常结束(SLDC)
0	1	0	复位外部状态中断
0	1	1	通道复位
1	0	0	遇到第一个字符时复位Rx中断
1	0	1	复位挂起的Rx中断
1	1	0	错误复位
1	1	1	中断返回(仅通道A)

D7	D6	
0	0	无效代码
0	1	复位Rx CRC校验器
1	0	复位Tx CRC发生器
1	1	复位CRC/SYNC送/发送锁存

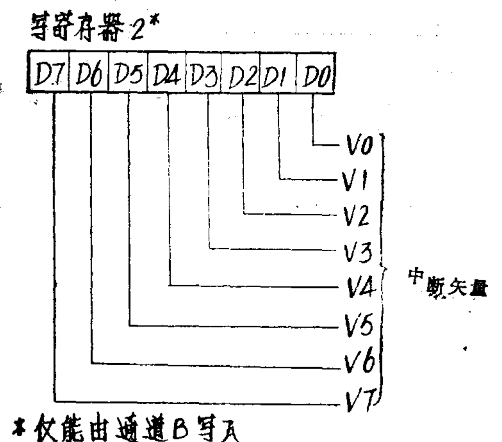
写寄存器 1

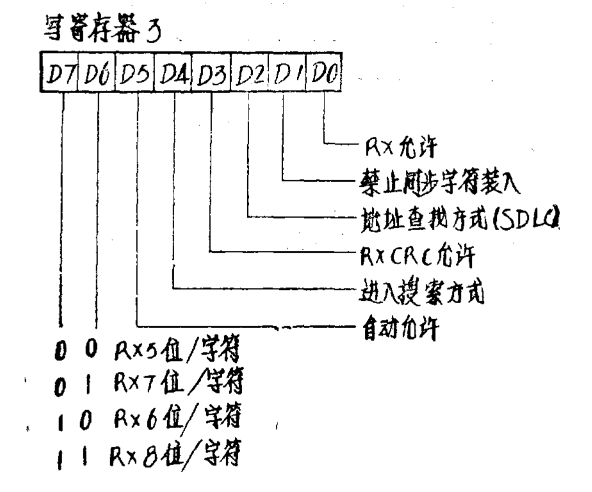
D7	D6	D5	D4	D3	D2	D1	D0

- D0 — 外部中断允许
- D1 — Tx中断允许
- D2 — 状态影响矢量

D4	D3	
0	0	Rx中断允许
0	1	仅遇到第一个字符出错时Rx中断
1	0	遇到所有Rx字符都中断(奇偶影响矢量)
1	1	遇到所有Rx字符都中断(奇偶不影响矢量)

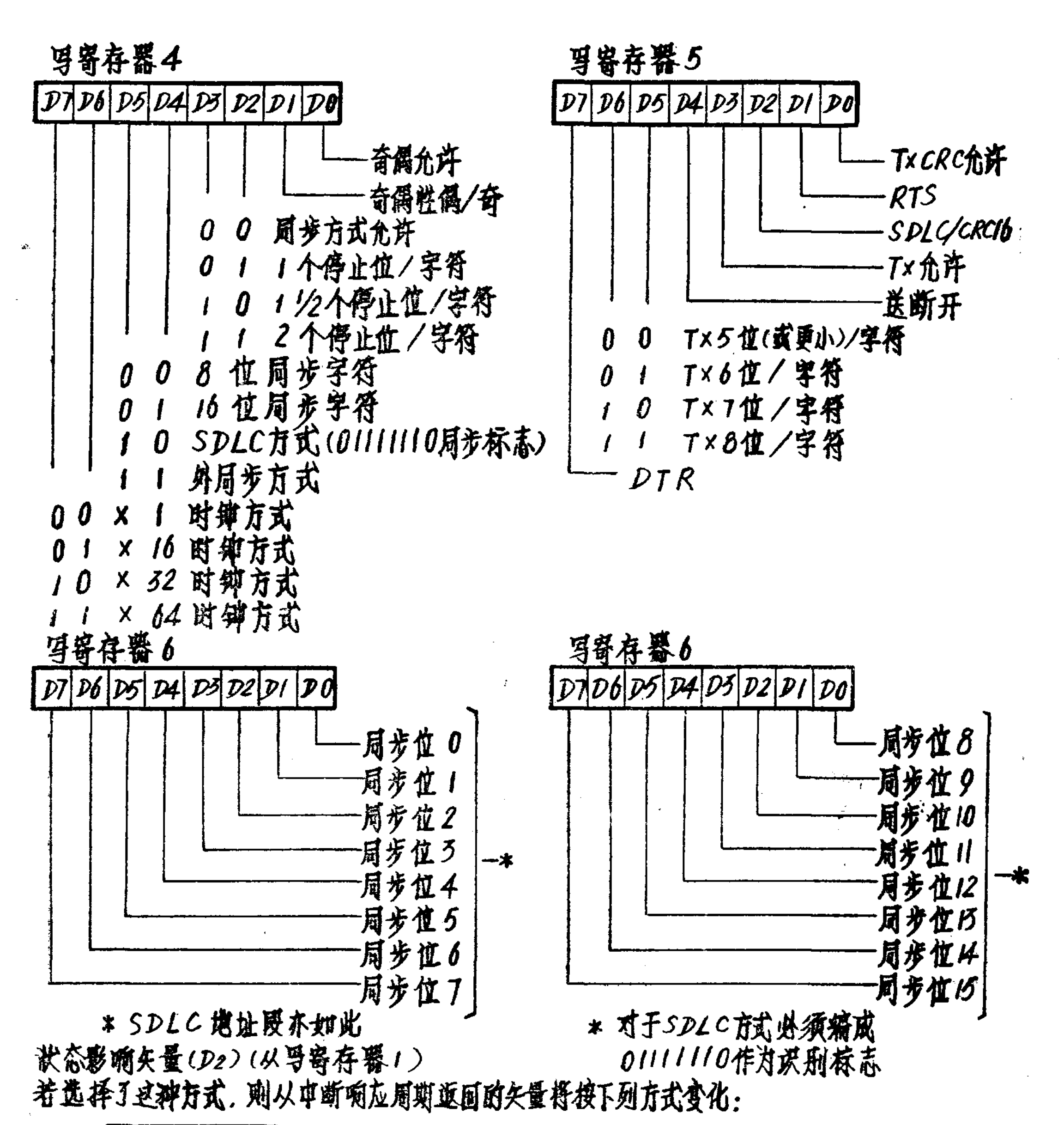
- D5 — WAIT/READY ON R/T
- D6 — WATI/FN/READY FN
- D7 — WAIT/READY 允许





D7	D6	D5	D4	D3	D2	D1	D0

SIO编 程 摘 要(续)



状态影响矢量(D_2)(从写寄存器1)

若选择了这种方式，则从中断响应周期返回的矢量将按下列方式变化：

	V3	V2	V1	
ChB	0	0	0	通道B发送缓冲器空
	0	0	1	通道B外部/状态改变
	0	1	0	通道B接收字符有效
	0	1	1	通道B特定接收条件
ChA	1	0	0	通道A发送缓冲器空
	1	0	1	通道A外部/状态改变
	1	1	0	通道A接收字符有效
	1	1	1	通道A特定接收条件

如果此位是0，则在中断矢量寄存器中编程的固定矢量被送回

ASCII(美国信息交换标准码)字符表

(7位码)

LSD \ MSD		0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0	0000	NUL	DLE	SP	0	@	P	`	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	″	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENG	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	′	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	•	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	]	k	
C	1100	FF	ES	′	<	L	′	l	|
D	1101	CR	GS	−	=	M	[	m	
E	1110	SO	RS	•	>	N	↑	n	~
F	1111	SI	VS	/	?	O	←	o	DEL

Z—80 单板机的输入输出接口应用
——光电输入及 D/A 输出通道

电工及计算机科学系　　吴报鑫　张华宋
上海长江电子计算机厂　　赵金荣

微型计算机工作时需要与外界交换信息。例如：它的工作程序要由外界输入，处理的结果又需要去控制外部设备动作或由外部设备显示、记录、描绘。接口技术能使这些信息交换得以实现。

接口技术要解决二方面的问题：一是如何使外部设备与微型计算机连接并传送数据与控制信息；另一是如何对 I/0 设备寻址，使 CPU 能选出某一特定的设备与之联系。

一、Z—80 单板机常用的接口器件

1. Z—80 并行 I/O 器件(PIO)

PIO 器件由锁存器、缓冲器、触发器及一些逻辑电路组成。它有二个8位口：口 A 口 B。每个口有二个寻址单元，一个是控制寄存器，一个是数据寄存器。PIO 有四种操作方式：方式 0 是带联络的输出方式；方式 1 是带联络的输入方式；方式 2 是带联络的双向 I/O(只有口 A 可工作在这一方式)；方式 3 是位控操作方式。PIO 的寻址由 $\overline{CE}$，C/D 和 B/A 三引脚共同完成。如表一所示。

表 1　　PIO 的选择逻辑

$\overline{CE}$ (A_7-A_2)	C/D (A_1)	B/A (A_0)	选中单元	地址
0	0	0	口 A 数据寄存器	PIOADD
0	0	1	口 B 数据寄存器	PIOADD+1
0	1	0	口 A 控制寄存器	PIOADD+2
0	1	1	口 B 控制寄存器	PIOADD+3

$\overline{CE}$ 是芯片允许(低电平有效)。C/D 选择数据寄存器或控制寄存器。B/A 选择口 A 或口 B。

在程序的初始化阶段必须一次对 PIO 的控制寄存器赋以功能。当作输入、输出方式工作时应赋以下三种控制字：①工作方式②中断矢量③中断允许控制字。程序的其余阶段只是使同 PIO 的数据寄存器。

2．8212 八位 I/O 接口器件

8212I/O 接口具备控制部件和器件选择逻辑，有一个包含三态输出缓冲器的 8 位锁存器及一个服务请求触发器，后者可用来向 CPU 申请中断，$\overline{DS_1}$、DS_2、MD 和 STB 四只引脚信号的不同组合使器件实现不同的控制逻辑。见表二。

表 2　　8212 控制逻辑

STB	MD	$\overline{(DS_1 \cdot DS_2)}$	数据输出等于
0	0	0	3　态
1	0	0	3　态
0	1	0	数据锁存器
1	1	0	数据锁存器
0	0	1	数据锁存器
1	0	1	数据输入
0	1	1	数据输入
1	1	1	数据输入

$\overline{DS_1}$ 和 DS_2 是器件选择。当 $\overline{DS_1}$ 为低电平，DS_2 是高电平时，器件被选中。这时输出缓冲器允许工作。

MD 是方式选择。当 MD 是高电平时是输出方式，输出缓冲器允许工作而通向数据锁存器的时钟端来源是选择逻辑($\overline{DS_1}$，DS_2)。当 MD 为低电平时是输入方式，输出缓冲器状态由器件选择逻辑($\overline{DS_1}$、DS_2)决定，而数据锁存器的时钟端来源是 STB 端输入。

STB 是选通作用。用作通向数据锁存器的时钟端，并同步地使服务请求触发器复位。

二、光电输入接口

微型计算机工作所需的程序可以通过键盘手工输入；对需反复使用的常用程序可固化在 ROM 中或写入 EPROM 中，也可以通过诸如磁带、纸带，磁盘等外部设备保存程序信息，待每次需用时调入计算机。

本文着重介绍纸带通过光电机将程序信息送入计算机的接口技术。

1．硬件接口

RDG—10 微型光电机以光电转换方式，将纸带上的代码转换成电脉冲信号送到主机。带速是 150 排孔/秒。可用 5 单位或 8 单位纸带。需 +5V 和 +12V 二种电源。对外接口是 25 芯插头。光电机对外接口，代码是高电平有效。CPU 给光电机的启动命令是低电平有效。插头脚号如表三所示。与 CPU 接口如图一所示。

表 3　　　　　　　　　　光电插头引脚

脚　　号	名　　称	脚　　号	名　　称
1	第一代码道	13	+12V
2	二	14	+5V
3	三	15	+5V
4	四	16	+5V
5	五	17	备　用
6	六	18	
7	七	19	
8	八	20	
9	中导孔	21	
10	启动命令	22	地　　线
11	+12V	23	
12	+12V	24	
		25	

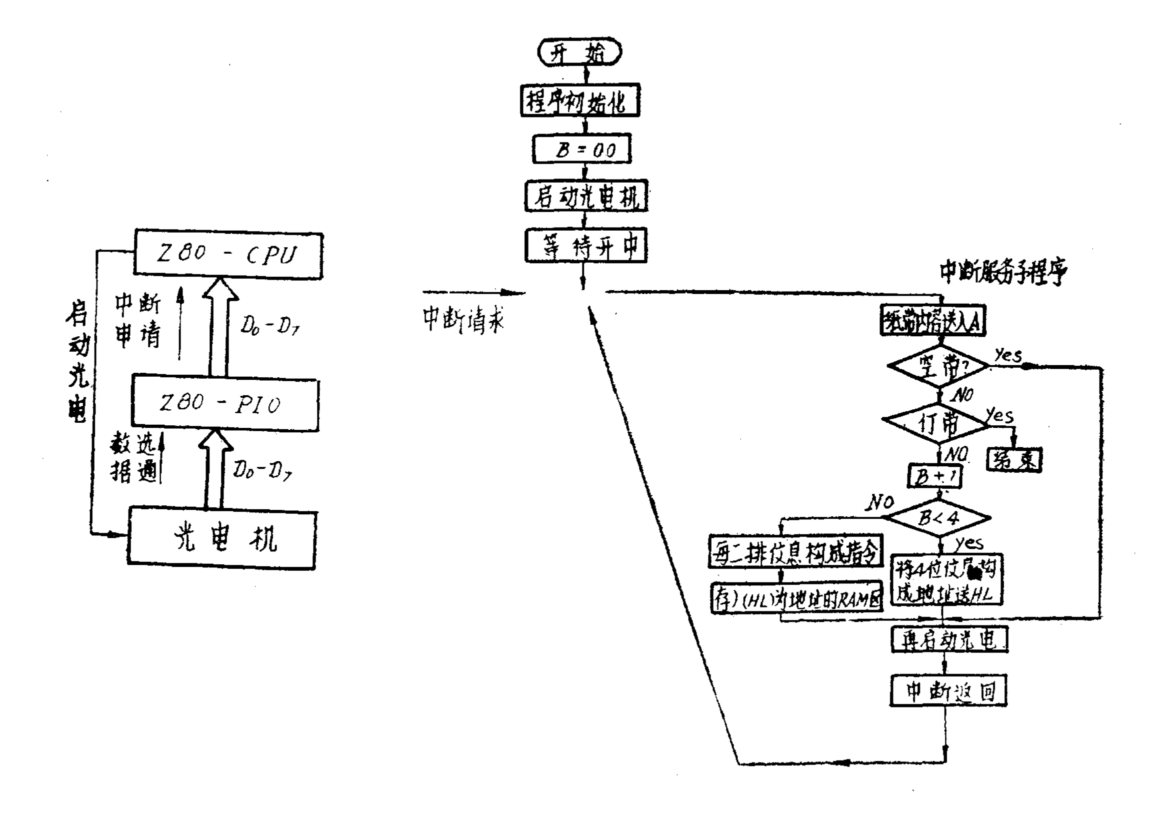


图 1　光电接口　　　　　　　　　　图 2　光电输入程序流程

2．光电输入的程序设计

光电纸带采用五单位的，最高位是数据信息标志，其余四位按 8421 编码。纸带穿孔

时，最前四排应是表示信息在 RAM 中的存放首址。接着每二排代表一条指令代码。中断请求信号由纸带中导孔脉冲提供。实现光电输入的程序流程图如图 2 所示。

三、D/A 通道的应用

在微型计算机内部，所有信息都是以数字量形式传送和处理的，而计算机所控制的对象，往往需要输入模拟量。例如：CRT 显示，X—Y 绘图仪以及由模拟电压输入的一些工业控制系统等，它们需要的输入量是模拟电压量。这就需要由 D/A 通道完成使数字信号变换成模拟信号，即 D/A 转换。

1. D/A 通道接口电路

目前芯片已有多种型号。现在 AD1408/AD1508 来说明 D/A 实现的方法。它是 16 脚双列直插式封装。它与 CPU 的接口可以通过 PIO 器件也可通过 8212 器件。这里以 8212 为例说明。如图二所示。

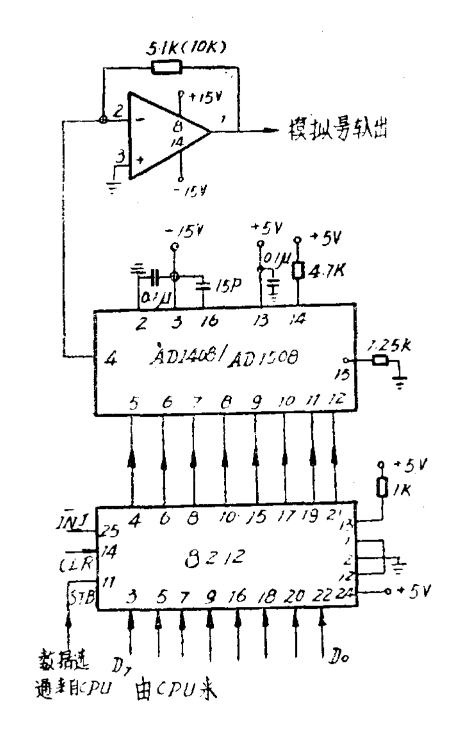


图 3 D/A 通道接口电路

开始
(HL)←首址
A←M
OUT(98),A
HL+1
A←M
OUT(9A),A
HL+1
NO
延时2
延时1

图 4 示波器显示流程

图三所示为单路 D/A 转换接口，若用于 X—Y 绘图仪或示波器显示图形，则需设双路 D/A 接口。转换出的模拟量一路送 X 方向，一路送 Y 方向。运算放大器的反馈电阻R接5.1K 时输出最大电压为 5V。R 为 10K 时，最大输出电压为 10V。

下面以示波器显示图形和 X—Y 绘图仪描绘曲线为例，说明 D/A 通道的软件设计。

2. D/A 通道的程序设计

当需要示波器或X—Y仪逐点描绘出某一图形时，可先将图形每个点的X、Y座标值列出，依次送入首址为规定值(用户自定)的存储区中，将X座标值存在前一单元，Y座标值存在后一单元，若所描图形能用函数表达式描述，则座标值可由程序相应建立。示波器显示的程序流程如图四所示。X—Y绘图仪描记曲线的程序流程如图五所示。

程序使存储在(HL)为首址的存储区中的数据不断反复循环送入CPU，光点即受控在屏幕上移动。经过适当的延时，使光点和图象能在屏幕上连续稳定地显示。

X—Y绘图仪图示波器显示的区别在于速度慢和需要自动按要求抬笔、落笔。对速度慢的问题可通过程序中适当延时加以解决，使送数速度与描记速度相适应。自动抬笔、落笔则还需通过绘图仪遥控端外加硬件实现。如图六所示。

由程序通过数据线最低位状态来控制D触发器$\overline{Q}$端状态，从而进一步控制光电耦合器的通断来使绘图仪抬笔或落笔。

由上所述可知，接口是微型计算机系统中的重要环节。微处理机用它自己的时钟以自己的速度进行工作。外部设备也有它的速度，要使二者配合起来协调工作，就需要通过接口来同步，这需要通过硬件和软件共同实现。

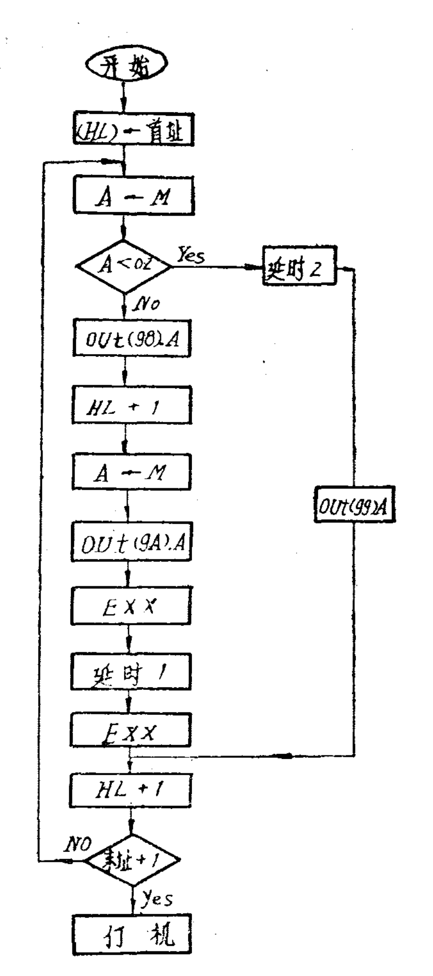


图5 X—Y绘图仪描记流程

Z—80微计算机有一组丰富的I/O指令。全部I/O指令使用8位设备地址。因而最多允许有256个输入口和输出口。在一台单板机控制多个输入输出设备时，通常采用中断方式。当外部设备准备送数或接受数据时，向CPU发出中断申请。CPU将按中断优先级别进入相应的中断服务子程序。当使用中断来启动I/O时，通常比普通的程序控制在硬件、软件方面都复杂一些。但却能提供CPU更快更直接地响应。

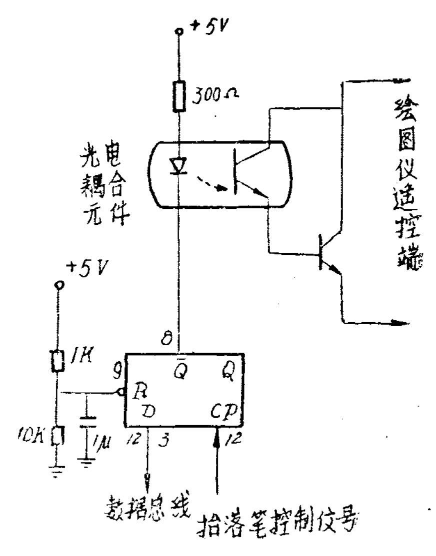


图6 绘图仪遥控接线图

Z—80 单板计算机功能扩充

上海长江电子计算机厂　　赵金荣
电工及计算机科学系　　吴报鑫　张华宋

采用大规模集成电路构成的单板计算机性能稳定可靠，重量轻，体积小，价格低廉，深受用户欢迎。但是，单板计算机的总线驱动能力较弱，随机配备的存贮器容量小，所以它所能带动的外部设备有限，软件的配备也较少，这样它的用途就受到一定的限制。如何进一步开发单板计算机的功能呢？使单板机既保持原有特色，又具有灵活的适应性呢？下面我们以Z—80单板机为例，介绍单板计算机的功能扩充。

一、总　线　驱　动

单板计算机通过数据总线、地址总线和控制总线把中央控制单元、存贮器和外部设备连接起来，实现对计算机本身及外围设备的控制。Z—80单板计算机的这三条总线是由CPU来直接驱动的，而CPU的驱动能力是有限的(能推动一个标准的TTL门)，假如存贮器的容量需要很大，被控制的对象很多时，CPU将出现过载，引起计算机工作的不稳定，甚至无法正常工作。因此，数据总线及地址总线的驱动，对于用单板机实现的自动控制系统来讲是必不可少的。为了保持单板机原有的特色，在保持原来单板机的总线外，将开辟一套新的总线——(数据总线)′、(地址总线)′，而构成双重总线方式。我们把原来的数据总线、地址总线和控制总线称为内部总线，(或称为单板机总线)，而把经驱动器驱动后的(数据总线)′、(地址总线)′称为外部总线，(或称为系统总线)。单板机总线与原来一样。对于系统总线来讲，(地址总线)′的传输是单方向的，由单板机送往外部设备；(数据总线)′的传输是双向的，可以由外部设备送往单板机，也可以由单板机送往外部设备，其传输方向由控制电路来实现。

单板机总线与系统总线的责职范围也有严格分工。单板机总线负责带动16K存贮器(地址码0000～3FFF)，一个PIO接口，一个CTC接口，一个键盘输入数字输出显示接口以及音频录音机接口。(共包括32个输入设备与32个输出设备，设备号80～9F)。系统总线负责带动再扩充的48K存贮器以及余下的所有外部设备口(包含224个输入设备及224个输出设备)，如图(一)所示。

经总线驱动后，整个单板计算机系统可以满足64K存贮器，带足256个输入设备及256个输出设备，为使用高级语言及扩充外部设备创造了有利条件。而单板机原有的功能(例MONITOR)，在功能扩充后的系统中完整地被保存下来。这样的单板机可大可小，可简单、可复杂，使用比原来更灵活，更方便了。实践运行表明，总线驱动后的单板机，使用更稳定，更可靠。

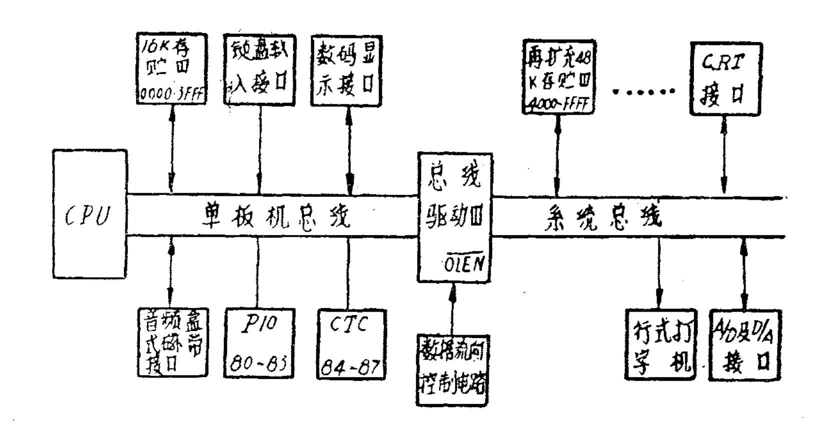


图1 Z—80单板计算机总线驱动框图

总线驱动器采用"四位双向总线驱动器"8216来实现。数据总线占用2片(D0～7)；地址总线有16位，只给占用3片，将A_{12}～A_{15}译成片选信号，供存贮器使用。(地址总线)′由单板机送向外部，当I/O操作时，其A_0～A_7提供外部设备号；当存贮器操作时，其A_0～A_{15}指明一个存贮器的读写地址。(数据总线)′是双向传送的，可以将待处理的数据由外部设备通过(数据总线)′送往单板机，也可以由单板机将处理好的数据通过(数据总线)′送往外部设备，实现控制。通过(数据总线)′，单板机可以对再扩充的48K存贮器进行读出或写入操作。

由于数据总线是双向传送的，因此必须设计一套"方向控制"电路，对8216的方向控制端($\overline{DIEN}$)实现控制。设计时按这样的原则来规定：

(1) 当$\overline{RD}=1$时，表示CPU不读数据，(数据总线)′的传输方向应该由单板机送向外部设备，即$\overline{DIEN}=1$。

(2) 当$\overline{RD}=0$时，表示CPU要读入数据，这时应从两种情况来分析：

(A) $\overline{MEMR}=0$，$\overline{IOR}=1$，表示CPU要向存贮器读入数据，那么要由这一次存贮器的读写地址而定。若读写地址是0000～3FFF，表示在0～16K范围内，由单板机总线来执行，系统总线不开放，所以$A_{15}=A_{14}=0$时，(数据总线)′的传输方向也由单板机送向外部。若读写地址是4000～FFFF时，表示在16K～64K范围内，由系统总线来执行，所以当$A_{15}\neq 0$或$A_{14}\neq 0$时，(数据总线)的传输方向应该由外部送往单板机，即$\overline{DIEN}=0$。

(B) $\overline{MEMR}=1$，$\overline{IOR}=0$，表示CPU要从外部设备读入数据，同样要由这一次I/O操作的设备号来确定。Z—80单板机上本身译成32个设备号，80，81，……9F。分别表示为$\overline{PS_0}$，$\overline{PS_1}$，……$\overline{PS_7}$。若PS_0～PS_7中有一个等于0时，表示这次操作由单板机总线执行，系统总线不开放，$\overline{DIEN}=1$。若Ps_0～Ps_7全部不等于0，即$\overline{Ps_0}\cdot\overline{Ps_1}\cdot\overline{Ps_2}\cdots\cdots\overline{Ps_7}=1$，表示这次操作由系统总线负责，$\overline{DIEN}=0$。

根据上面分析，我们可以列出真值表(表一)。

表 1　　　　方向控制电路真值表

$\overline{RD}$	$\overline{MEMR}$	A_{15}	A_{14}	$\overline{IOR}$	$\overline{Ps_0}$	$\overline{Ps_1}$	$\overline{Ps_2}$	$\overline{Ps_3}$	$\overline{Ps_4}$	$\overline{Ps_5}$	$\overline{Ps_6}$	$\overline{Ps_7}$	DIEN	说　明
1	×	×	×	×	×	×	×	×	×	×	×	×	1	CPU 不读数据
0	0	0	0	1	×	×	×	×	×	×	×	×	1	CPU 读存贮器
0	0	0	1	1	×	×	×	×	×	×	×	×	0	
0	0	1	0	1	×	×	×	×	×	×	×	×	0	
0	0	1	1	1	×	×	×	×	×	×	×	×	0	
0	1	×	×	0	0	×	×	×	×	×	×	×	1	CPU 读 I/O 口
0	1	×	×	0	×	0	×	×	×	×	×	×	1	
0	1	×	×	0	×	×	0	×	×	×	×	×	1	
0	1	×	×	0	×	×	×	0	×	×	×	×	1	
0	1	×	×	0	×	×	×	×	0	×	×	×	1	
0	1	×	×	0	×	×	×	×	×	0	×	×	1	
0	1	×	×	0	×	×	×	×	×	×	0	×	1	
0	1	×	×	0	×	×	×	×	×	×	×	0	1	
0	1	×	×	0	1	1	1	1	1	1	1	1	0	

注：表中×，表示可以是0，或是1。

由真值表，可以很容易地写出逻辑表达式，并画出有关的控制电路图。

数据总线与地址总线经驱动后，除了负载能力大大提高外，另一个优点是具有隔离作用。为了控制外围设备及扩充的存贮器，数据总线与地址总线必须从单板计算机引出，送往有关的接口电路。由于种种原因，不可避免地会产生对地或对电源短路，造成元器件损坏，若不经驱动器直接将总线引出，势必损坏 CPU、RAM、ROM、PIO、CTC 等贵重芯片。现在由于总线是经驱动电路(8216)后再送出，所以最多只损坏口上的驱动电路(8216)，损坏极微小。总线的这种驱动方式同样适用于其它类型的单板机或微型机。

二　存贮器的扩充

对一些控制系统来讲，4K 容量的 RAM 及 6K 容量的 EPROM 是不够的。单板机总线的驱动为存贮器的扩充创造了有利的条件。图二是 Z—80 单板机存贮器扩充的总体框图，通过扩充后，整机容量达 64K。采用 3 块 S—100 总线的 RAM 插件板来完成。

存贮器扩充分两种情况来处理：若系统需要的存贮器容量低于 16K 时，(地址空间为 0000～3FFF)，由单板机总线直接驱动；若总的存贮器容量大于 16K 时，则超出 16K 的存贮器(地址空间为 4000～FFFF)由系统总线负责驱动，而 16K 以内的存贮器(0000～3FFF)仍由单板机总线负责驱动。

扩充存贮器时，应注意以下几点：

(1) 由于 Z-80 单板机本身具有 EPROM 写入功能，在整个写入时间内(52ms)，CPU

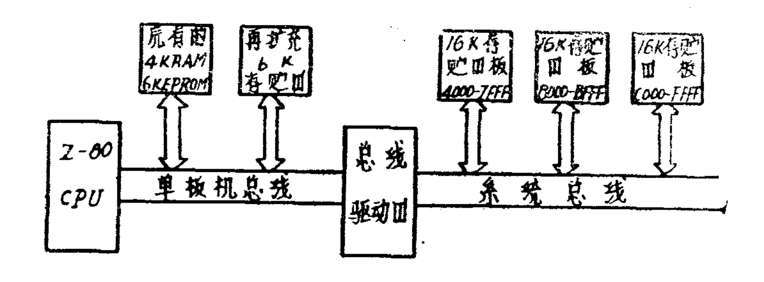


图2 Z-80单板机存贮器扩充总框图

处于等待状态，($\overline{WAIT}=0$)，暂停对动态存贮器的刷新，因此在扩充存贮器时，不宜采用动态存贮器，一般采用静态存贮器2114来实现。

(2) 扩充容量超出16K的这部分(4000～FFFF)，组装成三块S～100总线的插件板，按图二方式连接，即可投入运行。

(3) 对于扩充容量未超出16K的这部分存贮器，可按图三方式连接使用。

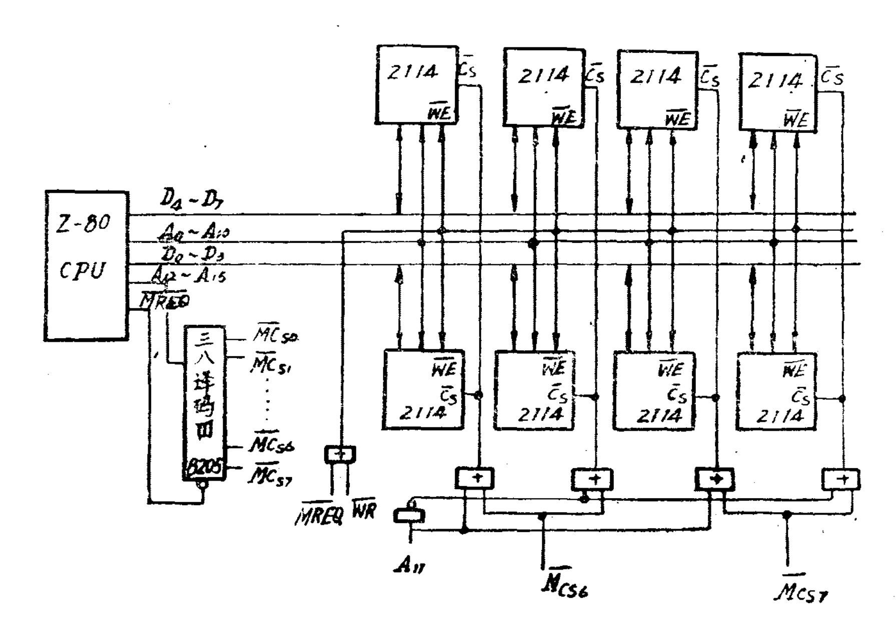


图3 Z—80单板机存贮器扩充到16K

经过总线驱动、存贮器容量扩充后的Z—80单板机，可以方便地连接上多种外部设备，从而形成Z—80单板计算机控制系统。

